Linux Professional Institute

Linux Essentials

Versione 1.6
Italiano

010

Table of Contents

ARGOMENTO 1: LA COMUNITÀ LINUX E UNA CARRIERA NELL'OPEN SOURCE . . . **1**
1.1 Evoluzione di Linux e Sistemi Operativi più Diffusi . . . **2**
1.1 Lezione 1 . . . 3
Introduzione . . . 3
Distribuzioni . . . 4
Sistemi Integrati . . . 5
Linux e Cloud . . . 7
Esercizi Guidati . . . 8
Esercizi Esplorativi . . . 9
Sommario . . . 10
Risposte agli Esercizi Guidati . . . 11
Risposte agli Esercizi Esplorativi . . . 13
1.2 Principali Applicazioni Open Source . . . **14**
1.2 Lezione 1 . . . 15
Introduzione . . . 15
Pacchetti Software . . . 15
Installazione di un Pacchetto . . . 16
Rimozione di un Pacchetto . . . 19
Applicazioni per Ufficio . . . 21
Browser Web . . . 22
Multimedia . . . 22
Programmi Server . . . 23
Condivisione dei Dati . . . 24
Amministrazione di Rete . . . 26
Linguaggi di Programmazione . . . 26
Esercizi Guidati . . . 29
Esercizi Esplorativi . . . 31
Sommario . . . 32
Risposte agli Esercizi Guidati . . . 33
Risposte agli Esercizi Esplorativi . . . 35
1.3 Software e Licenze Open Source . . . **36**
1.3 Lezione 1 . . . 37
Introduzione . . . 37
Definizione di Software Libero e di Software Open Source . . . 37
Licenze . . . 40
Modelli di Business nell'Open Source . . . 44
Esercizi Guidati . . . 46
Esercizi Esplorativi . . . 47

Sommario . . . 48
Risposte agli Esercizi Guidati . . . 49
Risposte agli Esercizi Esplorativi . . . 50
1.4 Competenze ICT - Lavorare con Linux . . . 52
1.4 Lezione 1 . . . 53
Introduzione . . . 53
Interfacce Utente in Linux . . . 54
Usi Industriali di Linux . . . 56
Problemi di Privacy nell'utilizzo di Internet . . . 57
Crittografia . . . 61
Esercizi Guidati . . . 64
Esercizi Esplorativi . . . 66
Sommario . . . 67
Risposte agli Esercizi Guidati . . . 68
Risposte agli Esercizi Esplorativi . . . 70
ARGOMENTO 2: TROVARE IL PROPRIO MODO DI OPERARE SU UN SISTEMA LINUX . . . 71
2.1 Nozioni di Base sulla Command Line . . . 72
2.1 Lezione 1 . . . 73
Introduzione . . . 73
Struttura della Command Line . . . 75
Tipi di Comportamento dei Comandi . . . 76
Quoting . . . 76
Esercizi Guidati . . . 80
Esercizi Esplorativi . . . 82
Sommario . . . 83
Risposte agli Esercizi Guidati . . . 84
Risposte agli Esercizi Esplorativi . . . 85
2.1 Lezione 2 . . . 86
Introduzione . . . 86
Variabili . . . 86
Gestire le Variabili . . . 87
Esercizi Guidati . . . 92
Esercizi Esplorativi . . . 93
Sommario . . . 94
Risposte agli Esercizi Guidati . . . 95
Risposte agli Esercizi Esplorativi . . . 96
2.2 Utilizzo della Command Line per Ottenere Aiuto . . . 98
2.2 Lezione 1 . . . 99
Introduzione . . . 99
Ottenere Aiuto dalla Command Line . . . 100

Trovare i File . . . 103
Esercizi Guidati . . . 105
Esercizi Esplorativi . . . 107
Sommario . . . 108
Risposte agli Esercizi Guidati . . . 109
Risposte agli Esercizi Esplorativi . . . 112
2.3 Utilizzo di Directory e Elenchi di File . . . 114
2.3 Lezione 1 . . . 115
Introduzione . . . 115
File e Directory . . . 115
Nomi di File e Directory . . . 116
Navigare nel Filesystem . . . 116
Percorsi Assoluti e Relativi . . . 118
Esercizi Guidati . . . 120
Esercizi Esplorativi . . . 122
Sommario . . . 123
Risposte agli Esercizi Guidati . . . 124
Risposte agli Esercizi Esplorativi . . . 127
2.3 Lezione 2 . . . 128
Introduzione . . . 128
Directory Home . . . 128
Il Percorso Relativo Speciale per la Directory Home . . . 130
Percorsi Relativi alla Home . . . 131
File e Directory Nascosti . . . 132
L'Opzione Formato di Elenco Lungo (Long List) . . . 133
Altre Opzioni di ls . . . 134
La Ricorsione in Bash . . . 135
Esercizi Guidati . . . 137
Esercizi Esplorativi . . . 139
Sommario . . . 140
Risposte agli Esercizi Guidati . . . 141
Risposte agli Esercizi Esplorativi . . . 143
2.4 Creazione, Spostamento ed Eliminazione di File . . . 144
2.4 Lezione 1 . . . 145
Introduzione . . . 145
Distinzione tra Maiuscole e Minuscole . . . 146
Creare Directory . . . 146
Creare i File . . . 148
Rinominare i File . . . 149
Spostare i File . . . 150

Eliminare File e Directory . . . 151
Copiare File e Directory . . . 153
Globbing . . . 155
Esercizi Guidati . . . 160
Esercizi Esplorativi . . . 162
Sommario . . . 163
Risposte agli Esercizi Guidati . . . 165
Risposte agli Esercizi Esplorativi . . . 168
ARGOMENTO 3: IL POTERE DELLA COMMAND LINE . . . 170
3.1 Archiviazione dei File sulla Command Line . . . 171
3.1 Lezione 1 . . . 172
Introduzione . . . 172
Strumenti di Compressione . . . 173
Strumenti di Archiviazione . . . 176
Gestire i File ZIP . . . 179
Esercizi Guidati . . . 181
Esercizi Esplorativi . . . 182
Sommario . . . 183
Risposte agli Esercizi Guidati . . . 185
Risposte agli Esercizi Esplorativi . . . 187
3.2 Ricerca ed Estrazione di Dati dai File . . . 188
3.2 Lezione 1 . . . 189
Introduzione . . . 189
Redirezione di I/O . . . 189
Pipe . . . 194
Esercizi Guidati . . . 196
Esercizi Esplorativi . . . 197
Sommario . . . 198
Risposte agli Esercizi Guidati . . . 199
Risposte agli Esercizi Esplorativi . . . 201
3.2 Lezione 2 . . . 202
Introduzione . . . 202
Cercare all'interno dei File con `grep` . . . 202
Espressioni Regolari . . . 203
Esercizi Guidati . . . 207
Esercizi Esplorativi . . . 208
Sommario . . . 209
Risposte agli Esercizi Guidati . . . 210
Risposte agli Esercizi Esplorativi . . . 212
3.3 Trasformare i Comandi in uno Script . . . 214

3.3 Lezione 1 . . . 215
Introduzione . . . 215
Stampare l’output . . . 215
Rendere uno Script Eseguibile . . . 216
Comandi e PATH . . . 216
Permessi di Esecuzione . . . 217
Definire un Interprete . . . 218
Variabili . . . 219
Uso delle Virgolette con le Variabili . . . 221
Argomenti . . . 222
Restituire il Numero di Argomenti . . . 224
Logica Condizionale . . . 224
Esercizi Guidati . . . 227
Esercizi Esplorativi . . . 229
Sommario . . . 230
Risposte agli Esercizi Guidati . . . 232
Risposte agli Esercizi Esplorativi . . . 234
3.3 Lezione 2 . . . 236
Introduzione . . . 236
Codici di Uscita . . . 237
Gestire più Argomenti . . . 239
Cicli For . . . 240
Usare le Espressioni Regolari per Eseguire il Controllo degli Errori . . . 243
Esercizi Guidati . . . 245
Esercizi Esplorativi . . . 247
Sommario . . . 248
Risposte agli Esercizi Guidati . . . 249
Risposte agli Esercizi Esplorativi . . . 251
ARGOMENTO 4: IL SISTEMA OPERATIVO LINUX . . . 252
4.1 Scelta di un Sistema Operativo . . . 253
4.1 Lezione 1 . . . 254
Introduzione . . . 254
Cos’è un Sistema Operativo . . . 254
Scegliere una Distribuzione Linux . . . 255
Sistemi Operativi non Linux . . . 259
Esercizi Guidati . . . 262
Esercizi Esplorativi . . . 264
Sommario . . . 265
Risposte agli Esercizi Guidati . . . 266
Risposte agli Esercizi Esplorativi . . . 268

4.2 Comprendere l'Hardware del Computer . . . **269**
4.2 Lezione 1 . . . 270
Introduzione . . . 270
Alimentatori . . . 271
Scheda Madre . . . 271
Memoria . . . 272
Processori . . . 273
Archiviazione . . . 276
Partitizioni . . . 277
Periferiche . . . 278
Driver e File di Dispositivo . . . 279
Esercizi Guidati . . . 281
Esercizi Esplorativi . . . 282
Sommario . . . 283
Risposte agli Esercizi Guidati . . . 284
Risposte agli Esercizi Esplorativi . . . 286
4.3 Dove Sono Memorizzati i Dati . . . **287**
4.3 Lezione 1 . . . 288
Introduzione . . . 288
I Programmi e la Loro Configurazione . . . 289
Il kernel Linux . . . 293
Dispositivi Hardware . . . 296
Memoria e Tipi di Memoria . . . 298
Esercizi Guidati . . . 300
Esercizi Esplorativi . . . 302
Sommario . . . 303
Risposte agli Esercizi Guidati . . . 305
Risposte agli Esercizi Esplorativi . . . 307
4.3 Lezione 2 . . . 308
Introduzione . . . 308
Processi . . . 308
Log di Sistema e Messaggistica di Sistema . . . 312
Esercizi Guidati . . . 318
Esercizi Esplorativi . . . 321
Sommario . . . 323
Risposte agli Esercizi Guidati . . . 325
Risposte agli Esercizi Esplorativi . . . 329
4.4 Il Tuo Computer in Rete . . . **331**
4.4 Lezione 1 . . . 332
Introduzione . . . 332

Rete Link Layer . . . 333
Rete IPv4 . . . 334
Rete IPv6 . . . 339
DNS . . . 342
Socket . . . 344
Esercizi Guidati . . . 346
Esercizi Esplorativi . . . 347
Sommario . . . 348
Risposte agli Esercizi Guidati . . . 349
Risposte agli Esercizi Esplorativi . . . 350
ARGOMENTO 5: SICUREZZA E PERMESSI SUI FILE . . . 352
5.1 Sicurezza di Base e Identificazione dei Tipi di Utente . . . 353
5.1 Lezione 1 . . . 354
Introduzione . . . 354
Account . . . 355
Ottenere Informazioni Sugli Utenti . . . 358
Cambiare Utente e Aumentare i Privilegi . . . 360
File di Controllo degli Accessi . . . 362
Esercizi Guidati . . . 369
Esercizi Esplorativi . . . 371
Sommario . . . 372
Risposte agli Esercizi Guidati . . . 374
Risposte agli Esercizi Esplorativi . . . 376
5.2 Creazione di Utenti e Gruppi . . . 378
5.2 Lezione 1 . . . 379
Introduzione . . . 379
Il File `/etc/passwd` . . . 380
Il File `/etc/group` . . . 381
Il File `/etc/shadow` . . . 381
Il File `/etc/gshadow` . . . 382
Aggiungere ed Eliminare Account Utenti . . . 383
La Directory Scheletro . . . 385
Aggiungere ed Eliminare Gruppi . . . 386
Il Comando `passwd` . . . 386
Esercizi Guidati . . . 388
Esercizi Esplorativi . . . 390
Sommario . . . 391
Risposte agli Esercizi Guidati . . . 392
Risposte agli Esercizi Esplorativi . . . 394
5.3 Gestione delle Autorizzazioni e delle Proprietà dei File . . . 397

5.3 Lezione 1 . . . 398
Introduzione . . . 398
Ricercare Informazioni su File e Directory . . . 398
E per le Directory? . . . 400
Visualizzare File Nascosti . . . 400
Comprendere i Tipi di File . . . 401
Comprendere i Permessi . . . 402
Modificare i Permessi dei File . . . 404
Modalità Simbolica . . . 405
Modalità Numerica . . . 406
Modificare la Proprietà dei File . . . 407
Ricercare i Gruppi . . . 408
Permessi Speciali . . . 409
Esercizi Guidati . . . 413
Esercizi Esplorativi . . . 415
Sommario . . . 416
Risposte agli Esercizi Guidati . . . 417
Risposte agli Esercizi Esplorativi . . . 420
5.4 Directory e File Speciali . . . 423
5.4 Lezione 1 . . . 424
Introduzione . . . 424
File Temporanei . . . 424
Comprendere i Link . . . 426
Esercizi Guidati . . . 431
Esercizi Esplorativi . . . 432
Sommario . . . 435
Risposte agli Esercizi Guidati . . . 436
Risposte agli Esercizi Esplorativi . . . 437
Imprint . . . 441

Linux
Professional
Institute

Argomento 1: La Comunità Linux e una Carriera nell'Open Source

1.1 Evoluzione di Linux e Sistemi Operativi più Diffusi

Obiettivi LPI di riferimento

Linux Essentials version 1.6, Exam 010, Objective 1.1

Peso

2

Aree di Conoscenza Chiave

- Distribuzioni
- Sistemi Integrati
- Linux nel Cloud

Di seguito è riportato un elenco parziale dei file, dei termini e dei comandi utilizzati

- Debian, Ubuntu (LTS)
- CentOS, openSUSE, Red Hat, SUSE
- Linux Mint, Scientific Linux
- Raspberry Pi, Raspbian
- Android

1.1 Lezione 1

Certificazione:	Linux Essentials
Versione:	1.6
Argomento:	1 La Comunità Linux e una Carriera nell'Open Source
Obiettivo:	1.1 Evoluzione di Linux e Sistemi Operativi più Diffusi
Lezione:	1 di 1

Introduzione

Linux è uno dei sistemi operativi più popolari. Il suo sviluppo è stato avviato nel 1991 da Linus Torvalds. Tale sistema operativo è stato ispirato da Unix, un altro sistema operativo sviluppato negli anni '70 dai Laboratori AT&T. Unix era pensato per i piccoli computer. In quegli anni i "piccoli" computer erano macchine che non necessitavano di un'intera sala con aria condizionata e costavano meno di un milione di dollari. Successivamente vennero considerate "piccole" le macchine che potevano essere sollevate da due persone. A quel tempo, un sistema Unix economico non era ancora disponibile per computer come quelli da ufficio, che tendevano ad essere basati sulla piattaforma x86. Pertanto Linus, che a quel tempo era studente, iniziò a implementare un sistema operativo di tipo Unix in grado di funzionare su quell'architettura.

Principalmente, Linux usa gli stessi principi e le idee di base di Unix, ma Linux stesso non contiene codice Unix, in quanto è un progetto indipendente. Linux non è supportato da una singola azienda ma da una comunità internazionale di programmatori. Poiché è liberamente disponibile, può essere usato da chiunque senza restrizioni.

Distribuzioni

Una *distribuzione* Linux è un pacchetto costituito da un *kernel* Linux ed un insieme di applicazioni che sono mantenute da una azienda o da una comunità di utenti. L'obiettivo di una distribuzione è quello di ottimizzare il kernel e le applicazioni in esecuzione sul sistema operativo per un particolare caso d'uso o gruppo di utenti. Le distribuzioni spesso includono strumenti specifici della distribuzione per l'installazione del software e l'amministrazione del sistema. Questo è il motivo per cui alcune distribuzioni vengono utilizzate principalmente per ambienti desktop dove devono essere facili da usare, mentre altre vengono utilizzate principalmente sui server per utilizzare le risorse disponibili nel modo più efficiente possibile.

Un altro modo per classificare le distribuzioni è quello di fare riferimento alla *famiglia di distribuzione* di appartenenza. Le distribuzioni della famiglia Debian usano il gestore di pacchetti `dpkg` per gestire il software che viene eseguito sul sistema operativo. I pacchetti che possono essere installati con il gestore di pacchetti sono gestiti da membri volontari della comunità della distribuzione. I manutentori utilizzano il formato di pacchetto `deb` per specificare come il software debba essere installato sul sistema operativo e come debba essere configurato di default. Proprio come una distribuzione, un pacchetto è un insieme di software con la relativa configurazione e documentazione che semplifica l'installazione, l'aggiornamento e l'utilizzo del software da parte dell'utente.

La distribuzione *Debian GNU/Linux* è la più grande distribuzione della famiglia Debian. Il progetto Debian GNU/Linux è stato lanciato da Ian Murdock nel 1993; oggi i volontari che lavorano al progetto sono migliaia. Debian GNU/Linux mira a fornire un sistema operativo molto affidabile. Inoltre promuove la visione di Richard Stallman di un sistema operativo che rispetti le libertà dell'utente di eseguire, studiare, distribuire e migliorare il software. Per questo motivo non fornisce di default alcun software proprietario.

Ubuntu è un'altra distribuzione basata su Debian degna di nota. Ubuntu è stata creata da Mark Shuttleworth e dal suo team nel 2004 con l'intento di offrire un ambiente desktop Linux facile da usare. Lo scopo di Ubuntu è quello di fornire un software libero per tutti in tutto il mondo e di ridurre il costo dei servizi professionali. La distribuzione prevede una nuova versione programmata ogni sei mesi con una versione di supporto a lungo termine ogni 2 anni.

Red Hat è una distribuzione Linux sviluppata e mantenuta dall'azienda software che porta lo stesso nome ed è stata acquisita da IBM nel 2019. La distribuzione Red Hat Linux è stata avviata nel 1994 e rinominata nel 2003 in *Red Hat Enterprise Linux*, spesso abbreviata in RHEL. Viene fornita alle aziende come soluzione di livello enterprise affidabile, supportata da Red Hat e dotata di un software che mira a facilitare l'utilizzo di Linux in ambienti server professionali. Alcuni dei suoi componenti richiedono abbonamenti o licenze a pagamento. Il progetto *CentOS* utilizza il codice sorgente liberamente disponibile di Red Hat Enterprise Linux e lo compila in una

distribuzione che è completamente gratuita, ma che, per contro, non prevede alcun supporto commerciale.

Sia RHEL che CentOS sono ottimizzati per l'uso in ambienti server. Il progetto *Fedora* è stato fondato nel 2003 e propone una distribuzione Linux rivolta ai computer desktop. Da allora Red Hat ha avviato e mantiene la distribuzione Fedora. Fedora è molto progressista e adotta nuove tecnologie molto rapidamente ed è talvolta considerato un banco di prova per le nuove tecnologie che in seguito potrebbero essere incluse in RHEL. Tutte le distribuzioni basate su Red Hat utilizzano il formato di pacchetto `rpm`.

L'azienda SUSE è stata fondata nel 1992 in Germania come fornitore di servizi Unix. La prima versione di *SUSE Linux* è stata rilasciata nel 1994. Nel corso degli anni SUSE Linux è diventata famosa soprattutto per il suo strumento di configurazione YaST. Questo strumento di configurazione permette agli amministratori di installare e configurare software e hardware, nonchè server e reti. Analogamente a RHEL, SUSE rilascia *SUSE Linux Enterprise Server*, che ne rappresenta la versione commerciale. È caratterizzata da rilasci poco frequenti ed è adatta per l'implementazione aziendale e di produzione. È distribuita sia come ambiente server che desktop, con pacchetti adatti allo scopo. Nel 2004 SUSE ha rilasciato il progetto *openSUSE* che ha concesso a sviluppatori e utenti l'opportunità di testare e sviluppare ulteriormente il sistema. La distribuzione openSUSE è liberamente scaricabile.

Nel corso degli anni sono state rilasciate distribuzioni indipendenti. Alcune di queste sono basate su Red Hat o Ubuntu; altre sono progettate per migliorare una specifica caratteristica di un sistema o di un hardware. Esistono distribuzioni costruite con funzionalità specifiche come *QubesOS*, un ambiente desktop molto sicuro, o *Kali Linux*, che fornisce un ambiente per testare le vulnerabilità software, utilizzato principalmente da coloro che effettuano test di penetrazione. Recentemente sono state progettate varie distribuzioni Linux leggerissime, per funzionare appositamente in contenitori Linux come Docker. Esistono anche distribuzioni create appositamente per componenti di sistemi integrati e persino dispositivi intelligenti.

Sistemi Integrati

I sistemi integrati sono una combinazione di hardware e software progettati per avere una funzione specifica all'interno di un sistema più grande. Solitamente fanno parte di altri dispositivi e aiutano a controllare i dispositivi stessi di cui fanno parte. I sistemi integrati si trovano in applicazioni di automazione, applicazioni mediche e persino militari. Grazie alla loro vasta gamma di applicazioni, sono stati sviluppati vari sistemi operativi basati sul kernel Linux per essere utilizzati nei sistemi integrati. Una parte significativa di dispositivi intelligenti ha al suo interno un sistema operativo basato su kernel Linux.

Pertanto, con i sistemi integrati viene fornito software integrato. Lo scopo di questo software è

quello di accedere all'hardware e renderlo utilizzabile. I principali vantaggi di Linux rispetto a qualsiasi altro software integrato proprietario risiedono nella compatibilità tra piattaforme diverse, nello sviluppo, nel supporto e nell'assenza di costi di licenza. Due dei più popolari progetti di software integrato sono Android, che viene utilizzato principalmente su telefoni cellulari di diversi fornitori, e Raspbian, che viene utilizzato principalmente su dispositivi Raspberry Pi.

Android

Android è principalmente un sistema operativo *mobile* sviluppato da Google. Android Inc. è stata fondata nel 2003 a Palo Alto, California. L'azienda ha inizialmente creato un sistema operativo pensato per funzionare su fotocamere digitali. Nel 2005, Google ha acquistato Android Inc. e lo ha sviluppato per renderlo uno dei più importanti sistemi operativi mobile.

La base di Android è una versione modificata del kernel Linux con software open source aggiuntivo. Il sistema operativo è sviluppato principalmente per dispositivi touchscreen, ma Google ha sviluppato anche versioni per TV e orologi da polso. Sono state sviluppate diverse versioni di Android per console di gioco, fotocamere digitali e PC.

Android è liberamente disponibile in ambiente open source come *Android Open Source Project* (AOSP). Google offre una serie di componenti proprietari oltre al nucleo open source di Android. Questi componenti includono applicazioni come Google Calendar, Google Maps, Google Mail, il browser Chrome ed il Google Play Store che facilita l'installazione delle applicazioni. La maggior parte degli utenti considera questi strumenti parte integrante della propria esperienza Android. Pertanto, quasi tutti i dispositivi mobile forniti con Android in Europa e America includono software proprietario di Google.

Android su dispositivi integrati presenta molti vantaggi. Il sistema operativo è intuitivo e facile da usare con una interfaccia utente grafica, ha una comunità di sviluppatori molto ampia, il che rende facile trovare aiuto per lo sviluppo. È inoltre supportato dalla maggior parte dei fornitori di hardware con un driver Android; quindi è facile ed economico creare un prototipo di un sistema completo.

Raspbian e il Raspberry Pi

Raspberry Pi è un computer a basso costo, dalle dimensioni di una carta di credito, che può funzionare come un computer desktop con funzionalità complete, ma che può essere anche utilizzato in un sistema Linux integrato. È sviluppato dalla Fondazione Raspberry Pi, un ente di beneficenza educativo con sede nel Regno Unito. Il suo scopo principale è quello di insegnare ai giovani a imparare a programmare e comprendere il funzionamento dei computer. Raspberry Pi può essere progettato e programmato per eseguire le attività o le operazioni desiderate che fanno parte di un sistema molto più complesso.

Le particolarità di Raspberry Pi includono una serie di pin *General Purpose Input-Output* (GPIO) che possono essere utilizzati per collegare dispositivi elettronici e schede di estensione. Ciò consente di utilizzare Raspberry Pi come piattaforma per lo sviluppo hardware. Sebbene inizialmente destinato a scopi didattici, Raspberry Pi è oggi utilizzato in vari progetti fai-da-te così come per la creazione di prototipi industriali durante lo sviluppo di sistemi integrati.

Raspberry Pi utilizza processori ARM. Vari sistemi operativi, tra cui Linux, funzionano su Raspberry Pi. Poiché Raspberry Pi non contiene un disco rigido, il sistema operativo viene avviato da una scheda di memoria SD. Una delle più importanti distribuzioni Linux per Raspberry Pi è *Raspbian*. Come suggerisce il nome, appartiene alla famiglia di distribuzione Debian. È personalizzato per essere installato sull'hardware Raspberry Pi e offre oltre 35000 pacchetti ottimizzati per questo ambiente. Oltre a Raspbian, esistono numerose altre distribuzioni Linux per Raspberry Pi, come, per esempio, Kodi, che trasforma il Raspberry Pi in un centro multimediale.

Linux e Cloud

Il termine *cloud computing* si riferisce a un modo standardizzato di gestire risorse informatiche, acquistandole presso un fornitore di cloud pubblico o eseguendo una cloud privata. Stando ai report del 2017, Linux gestisce il 90% del carico di lavoro della cloud pubblica. Ogni fornitore di servizi cloud, da *Amazon Web Services* (AWS) a *Google Cloud Platform* (GCP), offre diversi sistemi di tipo Linux. Addirittura Microsoft offre macchine virtuali basate su Linux nel suo cloud *Azure*.

Linux è di solito proposto come parte dell'offerta *Infrastructure as a Service* (IaaS). Le istanze IaaS sono macchine virtuali che vengono fornite in pochi minuti in cloud. Quando si avvia una istanza IaaS, viene scelta una immagine contenente i dati che verranno caricati sulla nuova istanza. I fornitori di servizi cloud offrono varie immagini contenenti installazioni pronte all'uso sia delle distribuzioni Linux più diffuse sia delle proprie versioni di Linux. L'utente sceglie un'immagine contenente la sua distribuzione preferita e poco dopo accede a una istanza cloud che esegue tale distribuzione. La maggior parte dei fornitori di servizi cloud aggiunge strumenti alle immagini per adattare l'installazione a una specifica istanza. Questi strumenti sono in grado, per esempio, di estendere i file system dell'immagine per adattarsi al reale disco rigido della macchina virtuale.

Esercizi Guidati

1. In che modo Debian GNU/Linux si differenzia da Ubuntu? Indica due aspetti.

2. Quali sono gli ambienti/piattaforme più comuni in cui viene utilizzato Linux? Indica tre differenti ambienti/piattaforme e una distribuzione che è possibile utilizzare per ciascuno di essi.

3. Stai pianificando di installare una distribuzione Linux in un nuovo ambiente. Indica quattro aspetti che dovresti tener presente quando scegli una distribuzione.

4. Indica tre dispositivi su cui è in esecuzione il sistema operativo Android, che non siano smartphone.

5. Spiega i tre principali vantaggi del cloud computing.

Esercizi Esplorativi

1. Considerando costi e prestazioni, quali sono le distribuzioni più adatte per un azienda che mira a ridurre i costi di licenza mantenendo le prestazioni ai massimi livelli? Spiega perchè.

2. Quali sono i principali vantaggi di Raspberry Pi e quali funzioni può avere in ambito lavorativo?

3. Quali distribuzioni offrono Amazon Cloud Services e Google Cloud? Indicane almeno tre in comune e due diverse.

Sommario

In questa lezione hai imparato:

- Quali distribuzioni ha Linux;
- Cosa sono i sistemi Linux integrati;
- Come sono utilizzati i sistemi Linux integrati;
- Differenti campi di utilizzo di Android;
- Diversi usi di un Raspberry Pi;
- Cos'è il Cloud Computing;
- Quale ruolo riveste Linux nel cloud computing.

 Questa opera è concessa in licenza CC BY-NC-ND 4.0. |

Risposte agli Esercizi Guidati

1. In che modo Debian GNU/Linux si differenzia da Ubuntu? Indica due aspetti.

   Ubuntu si basa su uno snapshot di Debian, motivo per cui ci sono molte caratteristiche simili tra loro. Tuttavia, ci sono altre differenze significative tra le due distribuzioni. La prima potrebbe essere l'applicabilità per i principianti. Ubuntu è raccomandato ai principianti per la sua facilità d'uso e, di contro, Debian è raccomandato per utenti più avanzati. La differenza principale è la complessità della configurazione dell'utente che Ubuntu non presenta durante il processo di installazione.

   Un'altra differenza potrebbe essere la stabilità. Debian è considerato più stabile di Ubuntu. Questo perché Debian riceve meno aggiornamenti che vengono testati in dettaglio e l'intero sistema operativo è più stabile. Invece Ubuntu consente all'utente di utilizzare le ultime versioni di software e tutte le nuove tecnologie.

2. Quali sono gli ambienti/piattaforme più comuni in cui viene utilizzato Linux? Indica tre differenti ambienti/piattaforme e una distribuzione che è possibile utilizzare per ciascuno di essi.

   Alcuni degli ambienti/piattaforme più comuni potrebbero essere smartphone, desktop e server. Su smartphone può essere utilizzato da distribuzioni come Android. Su desktop e server, può essere utilizzato da qualsiasi distribuzione adatta alle funzionalità di quella macchina, da Debian, Ubuntu a CentOS e Red Hat Enterprise Linux.

3. Stai pianificando di installare una distribuzione Linux in un nuovo ambiente. Indica quattro aspetti che dovresti tener presente quando scegli una distribuzione.

   Quando si sceglie una distribuzione, alcuni degli aspetti principali da considerare sono il costo, le prestazioni, la scalabilità, la stabilità e i requisiti hardware del sistema.

4. Indica tre dispositivi su cui è in esecuzione il sistema operativo Android, che non siano smartphone.

   Altri dispositivi su cui è in esecuzione Android sono le smart TV, i tablet, Android Auto e gli smartwatch.

5. Spiega i tre principali vantaggi del cloud computing.

   I principali vantaggi del cloud computing sono la flessibilità, la semplicità di ripristino e i bassi costi di utilizzo. I servizi basati su cloud sono semplici da implementare e ridimensionare, a seconda delle esigenze aziendali. Ha un grande vantaggio nelle soluzioni di backup e ripristino,

in quanto consente alle aziende di recuperare dai malfunzionamenti più velocemente e con minori ripercussioni. Inoltre, riduce i costi operativi, in quanto consente di pagare solo per le risorse utilizzate da un'azienda in base a un modello di abbonamento.

Risposte agli Esercizi Esplorativi

1. Considerando costi e prestazioni, quali sono le distribuzioni più adatte per un azienda che mira a ridurre i costi di licenza mantenendo le prestazioni ai massimi livelli? Spiega perchè.

   Una delle distribuzioni più adatte per essere utilizzate dalle aziende è CentOS. Questo perchè incorpora tutti i prodotti Red Hat, che sono anche utilizzati nel loro sistema operativo commerciale, pur essendo usufruibili gratuitamente. Allo stesso modo, le versioni Ubuntu LTS garantiscono il supporto per un periodo di tempo più lungo. Anche le versioni stabili di Debian GNU/Linux sono spesso utilizzate in ambienti aziendali.

2. Quali sono i principali vantaggi di Raspberry Pi e quali funzioni può avere in ambito lavorativo?

   Raspberry Pi è un dispositivo di ridotte dimensioni pur essendo in grado di funzionare come un normale computer. Inoltre è un dispositivo a basso costo ed è in grado di gestire traffico web e molte altre funzionalità. Può essere utilizzato come server, firewall e come scheda principale per robots o molti altri piccoli dispositivi.

3. Quali distribuzioni offrono Amazon Cloud Services e Google Cloud? Indicane almeno tre in comune e due diverse.

   Le distribuzioni comuni tra Amazon e Google Cloud Services sono Ubuntu, CentOS e Red Hat Enterprise Linux. Ogni fornitore di servizi cloud offre anche distribuzioni specifiche che gli altri fornitori non offrono. Amazon offre Amazon Linux e Kali Linux, mentre Google offre la possibilità di utilizzare FreeBSD e Windows Servers.

1.2 Principali Applicazioni Open Source

Obiettivi LPI di riferimento

Linux Essentials version 1.6, Exam 010, Objective 1.2

Peso

2

Aree di Conoscenza Chiave

- Applicazioni desktop
- Applicazioni server
- Linguaggi di sviluppo
- Strumenti di gestione dei pacchetti e repository

Di seguito è riportato un elenco parziale dei file, dei termini e dei comandi utilizzati

- OpenOffice.org, LibreOffice, Thunderbird, Firefox, GIMP
- Nextcloud, ownCloud
- Apache HTTPD, NGINX, MariaDB, MySQL, NFS, Samba
- C, Java, JavaScript, Perl, shell, Python, PHP
- dpkg, apt-get, rpm, yum

1.2 Lezione 1

Certificazione:	Linux Essentials
Versione:	1.6
Argomento:	1 La Comunità Linux e una Carriera nell'Open Source
Obiettivo:	1.2 Principali Applicazioni Open Source
Lezione:	1 di 1

Introduzione

Una applicazione è un programma per computer il cui scopo non è direttamente legato al funzionamento interno del computer, ma ai compiti eseguiti dall'utente. Le distribuzioni Linux offrono molti tipi di applicazioni per eseguire molteplici attività: applicazioni per ufficio, browser web, lettori ed editor multimediali, etc. Spesso c'è più di una applicazione o di uno strumento per eseguire un particolare lavoro: spetta all'utente scegliere l'applicazione più adatta alle proprie esigenze.

Pacchetti Software

Quasi ogni distribuzione Linux offre un insieme preinstallato di applicazioni predefinite. Oltre alle applicazioni preinstallate, una distribuzione ha un repository di pacchetti con un'ampia raccolta di applicazioni disponibili per l'installazione tramite il *package manager*. Sebbene le varie distribuzioni offrano più o meno le stesse applicazioni, esistono diversi sistemi di gestione dei pacchetti per le varie distribuzioni. Per esempio Debian, Ubuntu e Linux Mint, utilizzano gli strumenti `dpkg`, `apt-get` e `apt` per installare pacchetti software, generalmente indicati come *pacchetti DEB*. Distribuzioni come Red Hat, Fedora e CentOS utilizzano invece i comandi `rpm`, `yum`

e `dnf`, che a loro volta installano *pacchetti RPM*. Poiché il pacchetto di una applicazione è diverso per ciascuna famiglia di distribuzione, è molto importante installare i pacchetti dal repository corretto progettato per la specifica distribuzione. L'utente finale di solito non deve preoccuparsi di questi dettagli, poiché il gestore dei pacchetti della distribuzione sceglierà i pacchetti giusti, le dipendenze richieste e gli aggiornamenti futuri. Le dipendenze sono pacchetti aggiuntivi richiesti dai programmi. Per esempio, se una libreria fornisce funzioni per gestire le immagini JPEG utilizzate da vari programmi, è probabile che tale libreria venga "impacchettata" nello stesso pacchetto da cui dipendono tutte le applicazioni che usano la libreria.

I comandi `dpkg` e `rpm` operano su singoli file di pacchetto. In pratica, quasi tutte le attività di gestione dei pacchetti sono eseguite dai comandi `apt-get` o `apt` su sistemi che utilizzano pacchetti DEB o da `yum` o `dnf` su sistemi che utilizzano pacchetti RPM. Questi comandi funzionano con cataloghi di pacchetti, possono scaricare nuovi pacchetti e le loro dipendenze, verificare la presenza di versioni più recenti dei pacchetti installati.

Installazione di un Pacchetto

Supponi di aver sentito parlare di un comando chiamato `figlet` che stampa il testo ingrandito sul terminale e che tu voglia provarlo. Tuttavia, dopo aver eseguito il comando `figlet` si ottiene il seguente messaggio:

```
$ figlet
-bash: figlet: command not found
```

Questo probabilmente significa che il pacchetto non è installato sul tuo sistema. Se la tua distribuzione funziona con i pacchetti DEB, puoi cercare nei suoi repository usando `apt-cache search nome_pacchetto` o `apt search nome_pacchetto`. Il comando `apt-cache` è usato per cercare i pacchetti e per mostrare informazioni sui pacchetti disponibili. Il seguente comando cerca una qualunque occorrenza del termine "figlet" nei nomi e nelle descrizioni dei pacchetti:

```
$ apt-cache search figlet
figlet - Make large character ASCII banners out of ordinary text
```

La ricerca individua un pacchetto chiamato *figlet* che corrisponde al comando mancante. Le operazioni di installazione e rimozione di un pacchetto richiedono autorizzazioni speciali concesse solo all'amministratore di sistema: l'utente chiamato `root`. Sui sistemi desktop, gli utenti ordinari possono installare o rimuovere i pacchetti anteponendo il comando `sudo` ai comandi di installazione/rimozione. Ciò richiederà di digitare la password per procedere. Per i pacchetti DEB, l'installazione viene eseguita con il comando `apt-get install nome_pacchetto` o `apt`

`install nome_pacchetto`:

```
$ sudo apt-get install figlet
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  figlet
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
```

A questo punto il pacchetto verrà scaricato e installato sul sistema. Verranno scaricate e installate anche le dipendenze di cui il pacchetto ha eventualmente bisogno:

```
Need to get 184 kB of archives.
After this operation, 741 kB of additional disk space will be used.
Get:1 http://archive.raspbian.org/raspbian stretch/main armhf figlet armhf 2.2.5-2 [184 kB]
Fetched 184 kB in 0s (213 kB/s)
Selecting previously unselected package figlet.
(Reading database ... 115701 files and directories currently installed.)
Preparing to unpack .../figlet_2.2.5-2_armhf.deb ...
Unpacking figlet (2.2.5-2) ...
Setting up figlet (2.2.5-2) ...
update-alternatives: using /usr/bin/figlet-figlet to provide /usr/bin/figlet (figlet) in
auto mode
Processing triggers for man-db (2.7.6.1-2) ...
```

Al termine del download, tutti i file vengono copiati nelle posizioni corrette, viene eseguita qualsiasi configurazione aggiuntiva e il comando diviene disponibile:

```
$ figlet Awesome!
    _                                          _
   / \__      _____  ___  ___  _ __ ___   ___| |
  / _ \ \ /\ / / _ \/ __|/ _ \| '_ ` _ \ / _ \ |
 / ___ \ V  V /  __/\__ \ (_) | | | | | |  __/_|
/_/   \_\_/\_/ \___||___/\___/|_| |_| |_|\___(_)
```

Nelle distribuzioni basate su pacchetti RPM le ricerche vengono eseguite usando `yum search nome_pacchetto` o `dnf search nome_pacchetto`. Supponiamo che tu voglia visualizzare del testo in modo più divertente, accompagnato da una mucca disegnata a fumetto, ma non sei sicuro del pacchetto che può eseguire tale attività. Come nel caso dei pacchetti DEB, i comandi di ricerca

RPM accettano termini descrittivi:

```
$ yum search speaking cow
Last metadata expiration check: 1:30:49 ago on Tue 23 Apr 2019 11:02:33 PM -03.
=================== Name & Summary Matched: speaking, cow ====================
cowsay.noarch : Configurable speaking/thinking cow
```

Dopo aver trovato un pacchetto adatto nel repository, questo può essere installato con `yum install nome_pacchetto` o `dnf install nome_pacchetto`:

```
$ sudo yum install cowsay
Last metadata expiration check: 2:41:02 ago on Tue 23 Apr 2019 11:02:33 PM -03.
Dependencies resolved.
================================================================================
 Package         Arch           Version               Repository         Size
================================================================================
Installing:
 cowsay          noarch         3.04-10.fc28          fedora             46 k

Transaction Summary
================================================================================
Install  1 Package

Total download size: 46 k
Installed size: 76 k
Is this ok [y/N]: y
```

Ancora una volta, saranno scaricati e installati il pacchetto desiderato e tutte le sue possibili dipendenze:

```
Downloading Packages:
cowsay-3.04-10.fc28.noarch.rpm                  490 kB/s |  46 kB     00:00
================================================================================
Total                                            53 kB/s |  46 kB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing        :                                                        1/1
Installing       : cowsay-3.04-10.fc28.noarch                             1/1
```

```
Running scriptlet: cowsay-3.04-10.fc28.noarch                             1/1
Verifying        : cowsay-3.04-10.fc28.noarch                             1/1

Installed:
cowsay.noarch 3.04-10.fc28

Complete!
```

Il comando `cowsay` fa esattamente ciò che il suo nome suggerisce:

```
$ cowsay "Brought to you by yum"
 _________________________
< Brought to you by yum >
 -------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Sebbene possano sembrare inutili, i comandi `figlet` e `cowsay` forniscono un modo per attirare l'attenzione degli altri utenti su informazioni ritenute importanti.

Rimozione di un Pacchetto

Gli stessi comandi utilizzati per installare i pacchetti vengono utilizzati anche per rimuoverli. Tutti i comandi accettano la parola chiave `remove` per disinstallare un pacchetto installato: `apt-get remove nome_pacchetto` o `apt remove nome_pacchetto` per i pacchetti DEB; `yum remove nome_pacchetto` o `dnf remove nome_pacchetto` per i pacchetti RPM. Anche il comando `sudo` è necessario per eseguire la rimozione. Per esempio, per rimuovere il pacchetto *figlet* precedentemente installato da una distribuzione basata su DEB:

```
$ sudo apt-get remove figlet
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be REMOVED:
  figlet
0 upgraded, 0 newly installed, 1 to remove and 0 not upgraded.
After this operation, 741 kB disk space will be freed.
```

```
Do you want to continue? [Y/n] Y
```

Dopo aver confermato l'operazione, il pacchetto viene cancellato dal sistema:

```
(Reading database ... 115775 files and directories currently installed.)
Removing figlet (2.2.5-2) ...
Processing triggers for man-db (2.7.6.1-2) ...
```

Una procedura simile viene eseguita su un sistema basato su RPM. Per esempio, per rimuovere il pacchetto *cowsay* precedentemente installato da una distribuzione basata su RPM:

```
$ sudo yum remove cowsay
Dependencies resolved.
================================================================================
 Package          Arch             Version                Repository         Size
================================================================================
Removing:
 cowsay           noarch           3.04-10.fc28           @fedora            76 k

Transaction Summary
================================================================================
Remove  1 Package

Freed space: 76 k
Is this ok [y/N]: y
```

Allo stesso modo, viene richiesta una conferma e il pacchetto viene cancellato dal sistema:

```
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                        1/1
  Erasing          : cowsay-3.04-10.fc28.noarch                             1/1
  Running scriptlet: cowsay-3.04-10.fc28.noarch                             1/1
  Verifying        : cowsay-3.04-10.fc28.noarch                             1/1

Removed:
  cowsay.noarch 3.04-10.fc28
```

```
Complete!
```

I file di configurazione dei pacchetti rimossi sono conservati nel sistema, in modo che possano essere riutilizzati se il pacchetto verrà reinstallato in futuro.

Applicazioni per Ufficio

Le applicazioni per ufficio sono utilizzate per la modifica di file come testi, presentazioni, fogli di calcolo e altri formati comunemente utilizzati in un ambiente d'ufficio. Queste applicazioni sono solitamente organizzate in raccolte chiamate *office suites*.

Per molto tempo, la suite per ufficio più utilizzata in Linux è stata la *OpenOffice.org*. OpenOffice.org era una versione open source della *StarOffice suite* lanciata da *Sun Microsystems*. Pochi anni dopo Sun è stata acquisita da *Oracle Corporation*, che a sua volta ha trasferito il progetto alla *Apache Foundation* e OpenOffice.org è stata rinominata in *Apache OpenOffice*. Nel frattempo, un'altra suite per ufficio basata sullo stesso codice sorgente è stata rilasciata dalla *Document Foundation*: *LibreOffice*.

I due progetti hanno le stesse funzionalità di base e sono compatibili con i formati di documento di *Microsoft Office*. Tuttavia, il formato di documento preferito è l'*Open Document Format*, un formato di file completamente aperto e standardizzato ISO. L'uso di file ODF garantisce che i documenti possano essere trasferiti tra sistemi operativi e applicazioni di differenti fornitori, come lo stesso Microsoft Office. Le principali applicazioni offerte da OpenOffice/LibreOffice sono:

Writer
: Editor di testo

Calc
: Fogli di calcolo

Impress
: Presentazioni

Draw
: Disegno vettoriale

Math
: Formule matematiche

Base

Database

Sia LibreOffice sia Apache OpenOffice sono software open source, ma LibreOffice è concesso in licenza sotto LGPLv3 e Apache OpenOffice è concesso in licenza sotto Apache License 2.0. La distinzione di licenza implica che LibreOffice può incorporare le migliorie apportate da Apache OpenOffice, ma Apache OpenOffice non può incorporare le migliorie apportate da LibreOffice. Questo, insieme a una comunità di sviluppatori più attiva, è il motivo per cui la maggior parte delle distribuzioni adotta LibreOffice come suite predefinita per l'ufficio.

Browser Web

Per la maggior parte degli utenti, lo scopo principale di un computer è fornire accesso a Internet. Al giorno d'oggi le pagine Web possono funzionare come app a sé stanti, con il vantaggio di essere accessibili da qualsiasi luogo, senza la necessità di installare software aggiuntivo. Questo rende il browser web l'applicazione più importante del sistema operativo, almeno per l'utente medio.

TIP

Una delle migliori fonti per apprendere lo sviluppo web è MDN Web Docs, disponibile su https://developer.mozilla.org/. Gestito da Mozilla, il sito è ricco di tutorial per principianti e materiali di riferimento sulla maggior parte delle moderne tecnologie web.

I principali browser web in ambiente Linux sono *Google Chrome* e *Mozilla Firefox*. Chrome è un browser web gestito da Google ma è basato sul browser open source chiamato *Chromium*, che può essere installato usando il gestore di pacchetti della propria distribuzione ed è pienamente compatibile con Chrome. Gestito da Mozilla, un'organizzazione senza fini di lucro, Firefox è un browser le cui origini sono legate a Netscape, il primo popolare browser web ad adottare il modello open source. La Mozilla Foundation è profondamente coinvolta nello sviluppo di standard aperti che sono alla base del moderno web.

Mozilla sviluppa anche altre applicazioni, come il client di posta elettronica *Thunderbird*. Molti utenti scelgono di utilizzare la webmail invece di un'applicazione di posta elettronica dedicata, ma un client come Thunderbird offre funzionalità aggiuntive e si integra al meglio con le altre applicazioni desktop.

Multimedia

Rispetto alle applicazioni web disponibili, le applicazioni desktop sono ancora la migliore scelta per la creazione di contenuti multimediali. Attività multimediali come il rendering video spesso richiedono elevate quantità di risorse di sistema, che sono gestite al meglio da un'applicazione desktop locale. Di seguito sono elencate alcune delle applicazioni multimediali più popolari per

l'ambiente Linux ed i loro usi.

Blender

Un renderer 3D per creare animazioni. Blender può anche essere utilizzato per esportare oggetti 3D da stampare con una stampante 3D.

GIMP

Un editor di immagini completo, che può essere paragonato ad *Adobe Photoshop*, ma che ha i propri concetti e strumenti per lavorare con le immagini. GIMP può essere utilizzato per creare, modificare e salvare la maggior parte dei file bitmap, come JPEG, PNG, GIF, TIFF e molti altri.

Inkscape

Un editor di grafica vettoriale, simile a *Corel Draw* o *Adobe Illustrator*. Il formato predefinito di Inkscape è SVG, che è uno standard aperto per la grafica vettoriale. I file SVG possono essere aperti da un qualunque browser web e, per la loro natura di grafica vettoriale, possono essere utilizzati per layout flessibili di pagine web.

Audacity

Un editor audio. Audacity può essere utilizzato per filtrare, applicare effetti e conversioni a e tra diversi formati audio come MP3, WAV, OGG, FLAC, etc.

ImageMagick

ImageMagick è uno strumento a Command Line per convertire e modificare la maggior parte dei tipi di file immagine. Può anche essere utilizzato per creare documenti PDF da file immagine e viceversa.

Esistono anche molte applicazioni dedicate alla riproduzione multimediale. L'applicazione più popolare per la riproduzione di video è *VLC*, ma alcuni utenti preferiscono alternative come *smplayer*. Anche la riproduzione di musica locale ha molte opzioni, come *Audacious*, *Banshee* e *Amarok*, che possono anche gestire una raccolta di file audio locali.

Programmi Server

Quando un browser web carica una pagina da un sito web, in realtà si connette a un computer remoto e richiede determinate informazioni. In questo scenario, il computer che esegue il browser web viene chiamato *client* e il computer remoto viene chiamato *server*.

Il computer server, che può essere un normale computer desktop o un hardware specializzato, ha bisogno di un programma specifico per gestire ogni tipo di informazione che dovrà fornire. Per quanto riguarda la gestione delle pagine web, la maggior parte dei server in tutto il mondo

utilizza programmi server open source. Questo particolare programma server è chiamato *HTTP server* (HTTP sta per *Hyper Text Transfer Protocol*) e i più popolari sono *Apache*, *Nginx* e *lighttpd*.

Anche semplici pagine Web possono far fronte a molte richieste, che possono essere file ordinari, definiti contenuto statico, oppure contenuto dinamico generato da varie fonti. Il ruolo di un server HTTP è quello di raccogliere e inviare tutti i dati richiesti al browser, che quindi organizza il contenuto come definito dal documento HTML ricevuto (HTML significa *Hyper Text Markup Language*), e altri file di supporto. Pertanto il rendering di una pagina web comporta operazioni eseguite lato server e operazioni eseguite lato client. Entrambe le parti possono utilizzare script personalizzati per svolgere compiti specifici. Lato server HTTP, è abbastanza comune utilizzare il linguaggio di scripting PHP. JavaScript è il linguaggio di scripting utilizzato lato client (il browser Web).

I programmi server possono fornire informazioni di tutti i tipi. Non è raro avere un programma server che richiede informazioni fornite da altri programmi server. Questo è il caso in cui un server HTTP richiede informazioni fornite da un server di database.

Per esempio, quando viene richiesta una pagina dinamica, il server HTTP di solito interroga un database per raccogliere tutte le informazioni richieste ed invia il contenuto dinamico al client. Allo stesso modo, quando un utente si registra su un sito web, il server HTTP raccoglie i dati inviati dal client e li memorizza in un database.

Un database è un insieme organizzato di informazioni. Un server di database archivia i contenuti in modo formattato, consentendo di leggere, scrivere e collegare grandi quantità di dati in modo affidabile e ad alta velocità. I server di database open source sono utilizzati in molte applicazioni, non solo su Internet. Anche le applicazioni locali possono archiviare i dati connettendosi a un server di database locale. Il tipo più comune di database è il *database relazionale*, in cui i dati sono organizzati in tabelle predefinite. I database relazionali open source più popolari sono *MariaDB* (originato da *MySQL*) e *PostgreSQL*.

Condivisione dei Dati

Nelle reti locali, come quelle che si trovano negli uffici e nelle case, è auspicabile che i computer non solo possano accedere a Internet, ma siano anche in grado di comunicare tra loro. A volte un computer funziona da server, a volte lo stesso computer funziona da client. Ciò è necessario quando si desidera accedere ai file su un altro computer in rete, per esempio accedere a un file archiviato su un computer desktop da un dispositivo portatile senza il fastidio di copiare tale file su unità USB o simili.

Tra macchine Linux viene spesso utilizzato *NFS* (*Network File System*). Il protocollo NFS è il modo standard per condividere file system in reti costituite solamente da macchine Unix/Linux. Tramite

NFS, un computer può condividere una o più delle sue directory con computer specifici sulla rete, in modo che possano leggere e scrivere file in queste directory. NFS può anche essere utilizzato per condividere l'albero delle directory di un intero sistema operativo con i client che lo utilizzeranno per l'avvio. Questi computer, chiamati *thin client*, sono spesso utilizzati in reti di grandi dimensioni per evitare la manutenzione di ogni singolo sistema operativo su ogni macchina.

Se ci sono altri tipi di sistemi operativi collegati alla rete, si consiglia di utilizzare un protocollo di condivisione dei dati che può essere compreso da tutti. *Samba* soddisfa questo requisito. Samba implementa un protocollo per la condivisione di file sulla rete, che era stato originariamente realizzato per il sistema operativo Windows, ma che è ora compatibile con tutti i principali sistemi operativi. Con Samba, i computer della rete locale possono non solo condividere file, ma anche stampanti.

Su alcune reti locali l'autorizzazione data al login su una workstation è concessa da un server centrale, chiamato *controller di dominio*, che gestisce l'accesso a varie risorse locali e remote. Il controller di dominio è un servizio fornito da *Active Directory* di Microsoft. Le workstation Linux possono connettersi a un controller di dominio utilizzando Samba o un sottosistema di autenticazione chiamato *SSSD*. A partire dalla versione 4, Samba può anche funzionare come controller di dominio in reti eterogenee.

Se l'obiettivo è quello di implementare una soluzione di cloud computing in grado di fornire vari metodi di condivisione dati basati sul web, occorre considerare due alternative: *ownCloud* e *Nextcloud*. I due progetti sono molto simili poichè Nextcloud è uno spin-off di ownCloud, il che non è raro tra i progetti open source. Tali spin-off sono solitamente chiamati *fork*. Entrambi forniscono le stesse funzionalità di base: condivisione e sincronizzazione dei file, aree di lavoro collaborative, calendario, contatti e posta elettronica, il tutto tramite interfacce desktop, mobile e web. Nextcloud offre anche conferenze audio/video private, mentre ownCloud è più focalizzato sulla condivisione di file e sull'integrazione con software di terze parti. Molte altre funzionalità sono fornite sotto forma di plugin che possono essere attivati in un secondo momento, se necessario.

Sia ownCloud sia Nextcloud offrono una versione a pagamento con funzionalità extra e supporto esteso. Ciò che li rende diversi dalle altre soluzioni commerciali è la possibilità di installare Nextcloud o ownCloud su un server privato, gratuitamente, evitando di conservare i dati sensibili su un server sconosciuto. Poiché tutti i servizi dipendono dalla comunicazione HTTP e sono scritti in PHP, l'installazione deve essere eseguita su un server Web configurato in precedenza, come Apache. Se stai pensando di installare ownCloud o Nextcloud sul tuo server, assicurati di abilitare anche HTTPS per crittografare tutte le connessioni al tuo cloud.

Amministrazione di Rete

La comunicazione tra computer è possibile solo se la rete funziona correttamente. Normalmente, la configurazione della rete viene eseguita da un insieme di programmi in esecuzione sul router, responsabili della configurazione e del controllo della disponibilità della rete. A tal fine vengono utilizzati due servizi di rete essenziali: *DHCP* (*Dynamic Host Configuration Protocol*) e *DNS* (*Domain Name System*).

Il DHCP è responsabile dell'assegnazione di un indirizzo IP all'host quando viene collegato un cavo di rete o quando il dispositivo entra in una rete wireless. Durante la connessione a Internet, il server DHCP dell'ISP fornirà un indirizzo IP al dispositivo richiedente. Un server DHCP è molto utile anche nelle reti locali per fornire automaticamente indirizzi IP a tutti i dispositivi collegati. Se il DHCP non è configurato o se non funziona correttamente, potrebbe essere necessario configurare manualmente l'indirizzo IP di ciascun dispositivo connesso alla rete. Non è pratico impostare manualmente gli indirizzi IP in reti di grandi dimensioni o anche in piccole reti, motivo per cui la maggior parte dei router di rete ha un server DHCP preconfigurato di default.

L'indirizzo IP è necessario per comunicare con un altro dispositivo su una rete IP, ma i nomi di dominio come `www.lpi.org` hanno molte più probabilità di essere ricordati rispetto a un indirizzo IP come `203.0.113.165`. Tuttavia, il nome di dominio da solo non è sufficiente per stabilire la comunicazione attraverso la rete. Ecco perchè il nome di dominio deve essere tradotto in un indirizzo IP da un server DNS. L'indirizzo IP del server DNS è fornito dal server DHCP dell'ISP ed è utilizzato da tutti i sistemi connessi per tradurre i nomi di dominio in indirizzi IP.

Sia le impostazioni del DHCP sia quelle del DNS possono essere modificate entrando nell'interfaccia web fornita dal router. Per esempio, è possibile limitare l'assegnazione di indirizzi IP solo a dispositivi noti o associare un indirizzo IP fisso a macchine specifiche. È anche possibile modificare il server DNS predefinito fornito dall'ISP. Alcuni server DNS di terze parti, come quelli forniti da Google o da OpenDNS, possono talvolta fornire risposte più rapide e funzionalità aggiuntive.

Linguaggi di Programmazione

Tutti i programmi per computer (programmi client e server, applicazioni desktop e il sistema operativo stesso) sono realizzati utilizzando uno o più linguaggi di programmazione. I programmi possono essere un singolo file o un complesso sistema di centinaia di file, che il sistema operativo tratta come una sequenza di istruzioni che devono essere interpretate ed eseguite dal processore e da altri dispositivi.

Esistono numerosi linguaggi di programmazione per scopi molto diversi e i sistemi Linux ne forniscono un'ampia selezione. Poiché il software open source include anche il codice sorgente dei

programmi, i sistemi Linux offrono agli sviluppatori le condizioni perfette per comprendere, modificare o creare software in base alle proprie esigenze.

Ogni programma inizia come un file di testo, chiamato *codice sorgente*. Questo codice sorgente è scritto in un linguaggio più o meno leggibile dall'uomo e descrive ciò che il programma sta facendo. Un processore non può eseguire direttamente questo codice. Nei *linguaggi compilati*, il codice sorgente è quindi convertito in un *file binario* che può essere poi eseguito dal computer. Un programma chiamato *compilatore* è responsabile della conversione da codice sorgente a una forma eseguibile. Poiché il file binario compilato è specifico per un tipo di processore, potrebbe essere necessario ricompilare il programma per poterlo eseguire su un altro tipo di computer.

Nei *linguaggi interpretati* non è necessario che il programma sia stato precedentemente compilato. Invece, un *interprete* legge il codice sorgente ed esegue le sue istruzioni ogni volta che il programma viene eseguito. Questo rende lo sviluppo più facile e veloce, ma allo stesso tempo i programmi interpretati tendono a essere più lenti dei programmi compilati.

Ecco alcuni dei linguaggi di programmazione più popolari:

JavaScript

JavaScript è un linguaggio di programmazione utilizzato principalmente nelle pagine web. Le applicazioni JavaScript erano inizialmente molto semplici, come le routine di validazione dei moduli. A oggi JavaScript è considerato un linguaggio di prima classe e viene utilizzato per creare applicazioni molto complesse non solo sul web, ma anche su server e dispositivi mobili.

C

Il linguaggio di programmazione C è strettamente correlato con i sistemi operativi, in particolare Unix, ma viene utilizzato per scrivere qualsiasi tipo di programma per quasi ogni tipo di dispositivo. I grandi vantaggi di C sono la flessibilità e la velocità. Lo stesso codice sorgente scritto in C può essere compilato per essere eseguito su piattaforme e sistemi operativi diversi, con modifiche minime o nulle. Dopo essere compilato, tuttavia, il programma verrà eseguito solo sul sistema di destinazione.

Java

L'aspetto principale di Java è che i programmi scritti in questo linguaggio sono portatili, il che significa che lo stesso programma può essere eseguito su sistemi operativi differenti. Nonostante il nome, Java non è correlato a JavaScript.

Perl

Perl è un linguaggio di programmazione ampiamente utilizzato per elaborare contenuti testuali. Da grande importanza alle espressioni regolari, il che rende Perl un linguaggio adatto per il filtraggio e l'analisi del testo.

Shell

La shell, in particolare la shell Bash, non è solo un linguaggio di programmazione, ma anche un'interfaccia interattiva per eseguire altri programmi. I programmi shell, noti come *shell scripts*, possono automatizzare attività complesse o ripetitive nell'ambiente a Command Line.

Python

Python è un linguaggio di programmazione molto popolare tra gli studenti e i professionisti non direttamente coinvolti nell'informatica. Pur avendo caratteristiche avanzate, Python è un buon modo per iniziare a imparare la programmazione grazie al suo approccio di facile utilizzo.

PHP

PHP è maggiormente utilizzato come linguaggio di scripting lato server per generare contenuti per il web. La maggior parte delle pagine HTML online non sono file statici, ma contenuti dinamici generati dal server da varie fonti, come i database. I programmi PHP, a volte chiamati semplicemente pagine PHP o script PHP, sono spesso utilizzati per generare questo tipo di contenuti. Il termine LAMP deriva dalla combinazione di un sistema operativo Linux, un server Apache HTTP, un database MySQL (o MariaDB) e programmazione PHP. I server LAMP sono una soluzione molto popolare per eseguire server web. Oltre a PHP, tutti i linguaggi di programmazione descritti in precedenza possono essere utilizzati per implementare applicazioni di questo tipo.

C e Java sono linguaggi compilati. Per essere eseguito dal sistema, il codice sorgente scritto in C viene convertito in codice macchina binario, mentre il codice sorgente Java viene convertito in *bytecode* eseguito in uno speciale ambiente software chiamato *Java Virtual Machine*. JavaScript, Perl, Shell script, Python e PHP sono tutti linguaggi interpretati, anche chiamati *linguaggi di scripting*.

 Questa opera è concessa in licenza CC BY-NC-ND 4.0. |

Esercizi Guidati

1. Per ciascuno dei seguenti comandi, identifica se è associato al *sistema di gestione pacchetti Debian* o al *sistema di gestione pacchetti Red Hat*:

`dpkg`	
`rpm`	
`apt-get`	
`yum`	
`dnf`	

2. Quale comando può essere utilizzato per installare Blender su Ubuntu? Dopo l'installazione, come può essere eseguito il programma?

3. Quale applicazione della suite LibreOffice può essere utilizzata per lavorare con fogli di calcolo elettronici?

4. Quale browser web open source viene utilizzato come base per lo sviluppo di Google Chrome?

5. SVG è uno standard aperto per la grafica vettoriale. Qual è l'applicazione più popolare per la modifica dei file SVG nei sistemi Linux?

6. Per ciascuno dei seguenti formati di file, scrivi il nome di un'applicazione in grado di aprire e modificare il file corrispondente:

`png`	
`doc`	
`xls`	
`ppt`	
`wav`	

7. Quale pacchetto software consente la condivisione di file tra macchine Linux e Windows sulla rete locale?

Esercizi Esplorativi

1. Sai che i file di configurazione vengono mantenuti anche se il pacchetto associato viene rimosso dal sistema. Com'è possibile rimuovere automaticamente il pacchetto chiamato *cups* e i relativi file di configurazione da un sistema basato su DEB?

2. Supponiamo di avere molti file di immagini TIFF e di volerli convertire in JPEG. Quale pacchetto software potrebbe essere utilizzato per convertire tali file direttamente dalla Command Line?

3. Quale pacchetto software è necessario installare per poter aprire i documenti di Microsoft Word inviati da un utente Windows?

4. Ogni anno linuxquestions.org promuove un sondaggio sulle applicazioni Linux più popolari. Visita https://www.linuxquestions.org/questions/2018-linuxquestions-org-members-choice-awards-128/ e scopri quali applicazioni desktop sono più popolari tra gli utenti Linux esperti.

Sommario

In questa lezione hai imparato:

- I sistemi di gestione dei pacchetti utilizzati nelle principali distribuzioni Linux;
- Le applicazioni open source in grado di modificare i formati di file più diffusi;
- I programmi server alla base di molti importanti servizi Internet e di rete locale;
- I linguaggi di programmazione comuni e i loro usi.

Risposte agli Esercizi Guidati

1. Per ciascuno dei seguenti comandi, identifica se è associato al *sistema di gestione pacchetti Debian* o al *sistema di gestione pacchetti Red Hat*:

`dpkg`	sistema di gestione pacchetti Debian
`rpm`	sistema di gestione pacchetti Red Hat
`apt-get`	sistema di gestione pacchetti Debian
`yum`	sistema di gestione pacchetti Red Hat
`dnf`	sistema di gestione pacchetti Red Hat

2. Quale comando può essere utilizzato per installare Blender su Ubuntu? Dopo l'installazione, come può essere eseguito il programma?

   Il comando `apt-get install blender`. Il nome del pacchetto dovrebbe essere specificato in minuscolo. Il programma può essere eseguito direttamente dal terminale con il comando `blender` o selezionandolo dal menu delle applicazioni.

3. Quale applicazione della suite LibreOffice può essere utilizzata per lavorare con fogli di calcolo elettronici?

   Calc

4. Quale browser web open source viene utilizzato come base per lo sviluppo di Google Chrome?

   Chromium

5. SVG è uno standard aperto per la grafica vettoriale. Qual è l'applicazione più popolare per la modifica dei file SVG nei sistemi Linux?

   Inkscape

6. Per ciascuno dei seguenti formati di file, scrivi il nome di un'applicazione in grado di aprire e modificare il file corrispondente:

`png`	Gimp
`doc`	LibreOffice Writer
`xls`	LibreOffice Calc
`ppt`	LibreOffice Impress

`wav`	Audacity

7. Quale pacchetto software consente la condivisione di file tra macchine Linux e Windows sulla rete locale?

   Samba

Risposte agli Esercizi Esplorativi

1. Sai che i file di configurazione vengono mantenuti anche se il pacchetto associato viene rimosso dal sistema. Com'è possibile rimuovere automaticamente il pacchetto chiamato *cups* e i relativi file di configurazione da un sistema basato su DEB?

   `apt-get purge cups`

2. Supponiamo di avere molti file di immagini TIFF e di volerli convertire in JPEG. Quale pacchetto software potrebbe essere utilizzato per convertire tali file direttamente dalla Command Line?

   ImageMagick

3. Quale pacchetto software è necessario installare per poter aprire i documenti di Microsoft Word inviati da un utente Windows?

   LibreOffice o OpenOffice

4. Ogni anno linuxquestions.org promuove un sondaggio sulle applicazioni Linux più popolari. Visita https://www.linuxquestions.org/questions/2018-linuxquestions-org-members-choice-awards-128/ e scopri quali applicazioni desktop sono più popolari tra gli utenti Linux esperti.

   Browser: Firefox. Client di posta elettronica: Thunderbird. Media player: VLC. Editor di grafica raster: GIMP.

1.3 Software e Licenze Open Source

Obiettivi LPI di riferimento

Linux Essentials version 1.6, Exam 010, Objective 1.3

Peso

1

Aree di Conoscenza Chiave

- Filosofia Open Source
- Le licenze Open Source
- Free Software Foundation (FSF), Open Source Initiative (OSI)

Di seguito è riportato un elenco parziale dei file, dei termini e dei comandi utilizzati

- Copyleft, Permissive
- GPL, BSD, Creative Commons
- Free Software, Open Source Software, FOSS, FLOSS
- Modelli di business Open Source

1.3 Lezione 1

Certificazione:	Linux Essentials
Versione:	1.6
Argomento:	1 La Comunità Linux e una Carriera nell'Open Source
Obiettivo:	1.3 Software e Licenze Open Source
Lezione:	1 di 1

Introduzione

Sebbene i termini *software libero* e *software open source* siano ampiamente utilizzati, ci sono ancora alcune idee sbagliate sul loro significato. In particolare, il concetto di "libertà" necessita di un esame più approfondito. Partiamo dunque dalla definizione dei termini.

Definizione di Software Libero e di Software Open Source

Criteri di Definizione del Software Libero

Prima di tutto, la parola "libero" nel contesto di software libero non ha nulla a che fare con "gratuito", o, come riassume il fondatore della *Free Software Foundation* (FSF), Richard Stallman:

> Per capire il concetto, dovresti pensare a "libero" come in "libertà di parola", non come in "birra gratis" .
>
> — Richard Stallman, Cos'è il software libero?

Indipendentemente dal fatto che si debba pagare o meno per il software, ci sono quattro criteri che caratterizzano il software libero. Richard Stallman descrive questi criteri come "le quattro libertà essenziali", il conteggio delle quali parte da zero:

- "La libertà di eseguire il programma come desideri, per qualsiasi scopo (libertà 0)."

  Il dove, come e per quale scopo viene utilizzato il software non possono essere né prestabiliti né limitati.

- "La libertà di studiare come funziona il programma e di adattarlo alle tue esigenze (libertà 1). L'accesso al codice sorgente ne è una condizione preliminare."

  Ognuno può modificare il software in base alle proprie idee ed esigenze. Ciò a sua volta presuppone che il cosiddetto *codice sorgente*, ovvero tutti i file di cui è composto un software, debba essere disponibile in un formato leggibile dai programmatori. E, naturalmente, questo diritto si applica tanto a un semplice utente che potrebbe voler aggiungere una singola funzionalità, quanto alle società di software che realizzano codice complesso come i sistemi operativi per smartphone o il firmware dei router.

- "La libertà di ridistribuire le copie in modo da poter aiutare gli altri (libertà 2)."

  Questa libertà incoraggia esplicitamente ogni utente a condividere il software con altri. Si tratta quindi della massima distribuzione possibile e quindi della più ampia comunità possibile di utenti e sviluppatori che, sulla base di queste libertà, sviluppano e migliorano ulteriormente il software a beneficio di tutti.

- "La libertà di distribuire copie delle tue versioni modificate ad altri (libertà 3). In questo modo puoi dare all'intera comunità la possibilità di trarre beneficio dalle tue modifiche. L'accesso al codice sorgente è una condizione preliminare per questo."

  Non si tratta solo della distribuzione del software libero, ma anche della distribuzione del software libero *modificato*. Chiunque apporti modifiche al software libero ha il diritto di renderle disponibili ad altri. Se lo fai, sei obbligato a farlo anche liberamente, cioè non devi limitare le libertà originali durante la distribuzione del software, anche se lo hai modificato o esteso. Per esempio, se un gruppo di sviluppatori ha idee diverse rispetto agli autori originali sulla direzione che dovrebbe prendere uno specifico software, può creare il proprio ramo di sviluppo (chiamato *fork*) e continuare a sviluppare il software come un nuovo progetto. Ma, ovviamente, tutti gli obblighi associati a queste libertà rimangono in vigore.

L'enfasi sull'idea di libertà è anche coerente nella misura in cui ogni movimento di libertà è diretto *contro* qualcosa, vale a dire un rivale che sopprime le libertà postulate, che considera il software come proprietà e vuole tenerlo "chiuso". A differenza del software libero, tale software è

chiamato *proprietario*.

Software Open Source vs. Software Libero

Per molti, *software libero* e *software open source* sono sinonimi. L'abbreviazione usata frequentemente *FOSS* per *Free and Open Source Software* sottolinea questa comunanza. FLOSS, che sta per *Free/Libre and Open Source Software*, è un altro termine popolare, che enfatizza inequivocabilmente l'idea di libertà anche per altre lingue diverse dall'inglese. Tuttavia, se si considera l'origine e lo sviluppo di entrambi i termini, vale la pena differenziarli.

Il termine *software libero* con la definizione delle quattro libertà descritte in precedenza risale a Richard Stallman e al progetto GNU da lui fondato nel 1985, quasi 10 anni prima della nascita di Linux. Il termine "GNU is not Unix" descrive l'obiettivo in un batter d'occhio: GNU è nato come iniziativa per sviluppare una soluzione tecnicamente convincente — ovvero il sistema operativo Unix — da zero, per renderlo disponibile al grande pubblico e per migliorarlo continuamente. L'apertura del codice sorgente era solo una necessità tecnica e organizzativa per raggiungere questo obiettivo, ma nella sua immagine il movimento del software libero è ancora un movimento *sociale* e *politico*, alcuni aggiungono anche ideologico.

Con il successo di Linux, le possibilità di collaborazione di Internet e le migliaia di progetti e aziende emersi in questo nuovo universo software, l'aspetto sociale è passato sempre più in secondo piano. L'apertura del codice sorgente stesso è passata da requisito tecnico a caratteristica determinante: quando il codice sorgente era visibile, il software era considerato "open source". Le motivazioni sociali hanno lasciato il posto a un approccio più pragmatico per lo sviluppo del software.

Il software libero e il software open source si occupano delle stesse questioni, con gli stessi metodi e in una comunità mondiale di individui, progetti e aziende. Ma poiché si sono avvicinati partendo da direzioni diverse — una sociale e una tecnico-pragmatica — a volte ci sono dei conflitti. Questi conflitti sorgono quando i risultati del lavoro congiunto non corrispondono agli obiettivi originali di entrambi i movimenti. Ciò accade soprattutto quando il software apre i suoi codici sorgenti ma non rispetta nel contempo le quattro libertà del software libero, per esempio quando esistono limitazioni alla divulgazione, alla modifica, o connessioni con altri componenti software.

La *licenza* sotto cui il software è disponibile determina a quali condizioni è soggetto un software in termini di utilizzo, distribuzione e modifica. E poiché i requisiti e le motivazioni possono essere molto diversi, sono state create innumerevoli licenze diverse nell'area FOSS. Dato l'approccio molto più radicale del movimento del software libero, non sorprende che questo non riconosca molte licenze open source come "libere" e quindi le rifiuti. Il contrario difficilmente accade a causa dell'approccio open source molto più pragmatico.

Diamo una breve occhiata alla complessa area delle licenze.

Licenze

A differenza di un frigorifero o di un'auto, un software non è un prodotto *fisico*, ma un prodotto *digitale*. Pertanto, un'azienda non può effettivamente trasferire la proprietà di tale prodotto vendendolo e modificandone il possesso fisico — piuttosto, trasferisce i diritti d'uso a quel prodotto e l'utente accetta contrattualmente tali diritti d'uso. Nella licenza del software sono registrati quali sono e soprattutto quali *non* sono i diritti d'uso: diventa quindi comprensibile quanto siano importanti le normative in essa contenute.

Mentre i grandi fornitori di software proprietario, come Microsoft o SAP, hanno le proprie licenze che sono esattamente su misura per i loro prodotti, i sostenitori del software libero e open source sin dall'inizio hanno lottato per la chiarezza e la validità generale delle loro licenze, perché dopo tutto ogni utente dovrebbe essere in grado di comprenderle e, se necessario, utilizzarle personalmente per i propri sviluppi.

Tuttavia, non si dovrebbe nascondere che questo ideale di semplicità difficilmente può essere raggiunto perché troppe esigenze specifiche e accordi giuridici non sempre compatibili a livello internazionale si frappongono a questo. Per fare solo un esempio: le leggi sul diritto d'autore tedesche e americane sono fondamentalmente diverse. Secondo la legge tedesca, esiste una *persona* come *autore* (più precisamente: *Urheber*), la cui opera è la sua *proprietà intellettuale*. Sebbene l'autore possa concedere il permesso di utilizzare la sua opera, non può cedere o rinunciare alla sua paternità. Quest'ultimo punto non esiste nella legge americana. Anche qui c'è un autore (che però può anche essere un'azienda o un'istituzione), ma ha solo diritti di sfruttamento che può trasferire in parte o in toto e quindi dissociarsi completamente dalla sua opera. Una licenza valida a livello internazionale deve essere interpretata nel rispetto delle diverse leggi.

Di conseguenza ci sono numerose, e a volte molto diverse, licenze FOSS. Peggio ancora sono le diverse *versioni* di una licenza, o un mix di licenze (all'interno di un progetto, o anche quando si collegano più progetti) che possono causare confusione o persino controversie legali.

Sia i rappresentanti del software libero sia i sostenitori del movimento open source decisamente orientato al mercato hanno creato le proprie organizzazioni, che oggi sono in gran parte responsabili della formulazione delle licenze software secondo i loro principi e supportano i loro membri nella loro applicazione.

Copyleft

La già citata *Free Software Foundation* (FSF) ha formulato la *GNU General Public License* (GPL)

come una delle licenze più importanti per il software libero, licenza che viene utilizzata per molti progetti, per esempio il kernel Linux. Inoltre, ha rilasciato licenze con adattamenti specifici del caso, come la *GNU Lesser General Public License* (LGPL), che regola la combinazione di software libero con codice modificato in cui il codice sorgente modificato non deve essere rilasciato al pubblico, la *GNU Affero General Public License* (AGPL), che regola la vendita dell'accesso al software ospitato, o la *GNU Free Documentation License* (FDL), che estende i principi di libertà alla documentazione del software. Inoltre, la FSF formula raccomandazioni a favore o contro le licenze di terze parti e progetti affiliati come GPL-Violations.org indagano su sospette violazioni delle licenze libere.

La FSF chiama il principio secondo il quale una licenza libera si applica anche alle varianti modificate del software *copyleft* — in contrasto con il principio del copyright restrittivo che rifiuta. L'idea, quindi, è di trasferire i principi liberali di una licenza software nel modo più illimitato possibile a future varianti del software al fine di prevenire successive restrizioni.

Ciò che sembra ovvio e semplice, tuttavia, porta a notevoli complicazioni nella pratica, motivo per cui i critici spesso chiamano il principio del copyleft "virale", poiché viene trasmesso alle versioni successive.

Da quanto detto risulta, per esempio, che due componenti software che sono concesse in licenza sotto differenti licenze copyleft potrebbero non essere combinabili tra loro, poiché entrambe le licenze non possono essere trasferite contemporaneamente al prodotto successivo. Questo può essere applicato anche a diverse versioni della stessa licenza!

Per questo motivo le licenze o le versioni di licenze più recenti spesso non abbracciano più il copyleft in modo così rigoroso. La già menzionata *GNU Lesser General Public License* (LGPL) è in questo senso una agevolazione per poter combinare software libero con componenti "non liberi", come spesso accade con le così dette *librerie*. Le librerie contengono subroutine o routine, che a loro volta vengono utilizzate da altri programmi. Ciò porta alla situazione comune in cui il software proprietario chiama una di queste subroutine da una libreria gratuita.

Un altro modo per evitare conflitti di licenza è la *doppia licenza*, in cui un software è concesso in licenza sotto diverse licenze, per esempio una licenza gratuita e una licenza proprietaria. Un caso d'uso tipico è una versione gratuita di un software che potrebbe essere utilizzata solo quando rispetta le restrizioni del copyleft e una offerta alternativa che consente di ottenere il software con una licenza diversa che libera il licenziatario da determinate restrizioni in cambio di una quota in denaro che potrebbe essere utilizzata per finanziare lo sviluppo del software.

Dovrebbe quindi risultar chiaro che la scelta della licenza per i progetti software dovrebbe essere fatta con molta cautela, poiché da essa dipendono la collaborazione con altri progetti, la combinabilità con altri componenti e anche il design futuro del proprio prodotto. Il copyleft

presenta agli sviluppatori sfide speciali a tal proposito.

Open Source Definition e Licenze "Permissive"

Sul lato open source, è l'*Open Source Initiative* (OSI), fondata nel 1998 da Eric S. Raymond e Bruce Perens, che si occupa principalmente di questioni di licenza. Ha inoltre sviluppato una procedura standardizzata per verificare la conformità delle licenze software con la sua *Open Source Definition*. Sul sito web OSI è attualmente possibile trovare più di 80 licenze open source riconosciute.

Qui vengono anche elencate le licenze come "approvate OSI" che contraddicono esplicitamente il principio del copyleft, in particolare il gruppo di *licenze BSD*. La *Berkeley Software Distribution* (BSD) è una variante del sistema operativo Unix originariamente sviluppata presso l'Università di Berkeley, variante che in seguito ha dato origine a progetti gratuiti come *NetBSD*, *FreeBSD* e *OpenBSD*. Le licenze alla base di questi progetti sono spesso indicate come *permissive*. Contrariamente alle licenze copyleft non hanno lo scopo di stabilire i termini di utilizzo delle varianti modificate. Piuttosto, la massima libertà dovrebbe aiutare il software a essere distribuito il più ampiamente possibile, lasciando che siano gli editor del software da soli a decidere come procedere con le modifiche — se, per esempio, rilasciarle a loro volta o trattarle come sorgenti chiusi e distribuirle commercialmente.

La *Licenza BSD a 2 clausule*, chiamata anche *Licenza BSD Semplificata* o *Licenza FreeBSD*, dimostra quanto possano essere succinte tali licenze permissive. Oltre alla clausola di responsabilità standard, che protegge gli sviluppatori da richieste di risarcimento derivanti da danni causati dal software, la licenza è costituita solo dalle due seguenti regole:

> Sono consentiti la ridistribuzione e l'uso in forma di codice sorgente e in forma binaria, con o senza modifiche, a patto che siano rispettate le seguenti condizioni:
>
> Le ridistribuzioni del codice sorgente devono conservare la nota di copyright sopra riportata, questa lista di condizioni e la seguente dichiarazione di non responsabilità.
>
> Le ridistribuzioni in forma binaria devono riprodurre la nota di copyright sopra riportata, questa lista di condizioni e la seguente dichiarazione di non responsabilità nella documentazione e/o in altri materiali forniti con la distribuzione.

Creative Commons

La concezione di sviluppo vincente di FLOSS e il progresso tecnologico associato hanno portato a tentativi di trasferire il principio dell'open source ad altre aree non tecniche. La preparazione e la messa a disposizione della conoscenza, così come la cooperazione creativa nella risoluzione di compiti complessi, sono ora considerate come una prova del principio open source, esteso e

relativo ai diversi ambiti di applicazione.

Da qui la necessità di creare basi affidabili anche in questi ambiti, che consentano di condividere ed elaborare i risultati lavorativi. Poiché le licenze software disponibili erano poco adatte a questo, ci furono numerosi tentativi di tramutare specifiche necessità, dal lavoro scientifico alle opere d'arte digitalizzate "nello spirito dell'open source", in licenze altrettanto utili.

L'iniziativa di gran lunga più importante di questo tipo oggi è *Creative Commons* (CC), che si descrive brevemente nel seguente modo:

> Creative Commons è un'organizzazione globale senza scopo di lucro che consente la condivisione e il riutilizzo della creatività e della conoscenza attraverso la fornitura di strumenti legali gratuiti.
>
> — https://creativecommons.org/faq/#what-is-creative-commons-and-what-do-you-do

Con Creative Commons, il fulcro dell'assegnazione dei diritti passa dal distributore all'autore. Un esempio: nell'editoria tradizionale, un autore di solito trasferisce tutti i diritti di pubblicazione (stampa, traduzione, etc.) a un editore, che a sua volta garantisce la migliore distribuzione possibile dell'opera. I canali di distribuzione di Internet significativamente mutati mettono ora l'autore nella posizione di esercitare molti di questi diritti di pubblicazione e di decidere autonomamente come deve essere utilizzato il loro lavoro. Creative Commons offre l'opportunità di determinare questo in modo semplice e legalmente affidabile, ma vuole ottenere anche di più: gli autori sono incoraggiati a rendere disponibili le proprie opere come contributo a un processo generale di scambio e cooperazione. A differenza del copyright tradizionale, che conferisce all'autore tutti i diritti che può trasferire ad altri secondo necessità, Creative Commons adotta l'approccio opposto: l'autore mette il suo lavoro a disposizione della comunità, ma può scegliere tra una serie di caratteristiche quelle che devono essere tenute in considerazione quando si utilizza l'opera — più caratteristiche sceglie, più restrittiva è la licenza.

Quindi il principio "Scegli una licenza" di CC chiede a un autore passo dopo passo le singole proprietà e genera la licenza consigliata, che l'autore può assegnare per ultimo all'opera come testo ed icona.

Per una migliore comprensione, ecco una panoramica delle sei possibili combinazioni e licenze offerte da CC:

CC BY ("Attribuzione")

La licenza gratuita che consente a chiunque di modificare e distribuire l'opera purché ne citi l'autore.

CC BY-SA ("Attribuzione-Condividi allo stesso modo")

Come CC BY, tranne per il fatto che l'opera modificata può essere distribuita solo sotto la stessa licenza. Il principio ricorda il copyleft, perché anche qui la licenza è "ereditata".

CC BY-ND ("Attribuzione-Non Opere Derivate")

Come CC BY, tranne per il fatto che l'opera può essere distribuita solo senza modifiche.

CC BY-NC ("Attribuzione-Non Commerciale")

L'opera può essere modificata e distribuita nominando l'autore, ma solo per scopi non commerciali.

CC BY-NC-SA ("Attributione-Non Commerciale-Condividi allo stesso modo")

Come BY-NC, tranne per il fatto che l'opera può essere condivisa solo alle stesse condizioni (cioè una licenza simile al copyleft).

CC BY-NC-ND ("Attribuzione-Non Commerciale-Non Opere Derivate")

La licenza più restrittiva: la distribuzione dell'opera è consentita nominando l'autore, ma solo senza modifiche e per scopi non commerciali.

Modelli di Business nell'Open Source

A posteriori il successo del FLOSS si traduce in un movimento di base di idealisti tecnofili che, indipendentemente dai vincoli economici e liberi da dipendenze monetarie, mettono il proprio lavoro al servizio del grande pubblico. Allo stesso tempo, in ambiente FLOSS sono state create aziende per un valore di miliardi; un esempio rappresentativo è la società americana *Red Hat* fondata nel 1993 con un fatturato annuo di oltre 3 miliardi di dollari (2018), rilevata dal colosso IT IBM nel 2018.

Diamo quindi un'occhiata alla tensione tra la distribuzione libera e per lo più gratuita di software di alta qualità e i modelli di business per i suoi creatori, perché una cosa dovrebbe essere chiara: anche gli innumerevoli sviluppatori, altamente qualificati, di software libero, devono guadagnare denaro, e l'ambiente FLOSS, in origine puramente non commerciale, deve quindi sviluppare modelli di business sostenibili al fine di preservare il proprio cosmo.

Un approccio comune, soprattutto per i progetti più grandi in fase iniziale, è il cosiddetto *crowdfunding*, ovvero la raccolta di donazioni in denaro tramite una piattaforma come *Kickstarter*. In cambio, i donatori ricevono un bonus prestabilito dagli sviluppatori in caso di successo, ovvero se vengono raggiunti gli obiettivi definiti in precedenza, sia che si tratti di accesso illimitato al prodotto o di funzionalità speciali.

Un altro approccio è la *doppia licenza*: il software libero viene offerto in parallelo con una licenza

più restrittiva o addirittura proprietaria, che a sua volta garantisce al cliente servizi più estesi (tempi di risposta in caso di errori, aggiornamenti, versioni per determinate piattaforme, etc.). Un esempio tra i tanti è *ownCloud*, che è stato sviluppato sotto licenza GPL e offre ai clienti aziendali una "Business Edition" sotto licenza proprietaria.

Prendiamo anche ownCloud come esempio di un altro modello di business FLOSS molto diffuso: i servizi professionali. Molte aziende non dispongono delle conoscenze tecniche interne necessarie per configurare e utilizzare software complessi e critici in modo affidabile e, soprattutto, sicuro. Ecco perché acquistano servizi professionali come consulenza, manutenzione o assistenza tecnica direttamente dal produttore. Anche gli aspetti di responsabilità giocano un ruolo in questa decisione, poiché l'azienda trasferisce i rischi operativi al produttore.

Se un software riesce a diventare popolare e di successo nel suo campo, ci sono possibilità di monetizzazione periferica come il merchandising o i certificati che i clienti acquisiscono e che quindi indicano il loro stato speciale nell'utilizzo di quel software. Per esempio, la piattaforma di apprendimento *Moodle* offre la certificazione dei formatori, che documentano le proprie conoscenze ai potenziali clienti, e questo è solo un esempio tra molti altri.

Software as a Service (SaaS) è un altro modello di business, soprattutto per le tecnologie basate sul web. In questo caso, un fornitore di servizi cloud esegue un software come per esempio un Customer Relationship Management (CRM) o un Content Management System (CMS) sui propri server e garantisce ai propri clienti l'accesso all'applicazione installata. Ciò consente ai clienti di risparmiare l'installazione e la manutenzione del software. In cambio, il cliente paga per l'utilizzo del software in base a vari parametri, per esempio il numero di utenti. Disponibilità e sicurezza giocano un ruolo importante come fattori critici per l'azienda.

Infine, ma cosa non di certo meno importante, è particolarmente comune nei progetti più piccoli il modello di sviluppo di estensioni specifiche su richiesta del cliente all'interno del software libero. Di solito spetta al cliente decidere cosa fare con queste estensioni, ovvero se rilasciarle o tenerle "chiuse" come parte del proprio modello di business.

Una cosa dovrebbe essere oramai chiara: sebbene il software libero sia di solito disponibile gratuitamente, nel suo ambiente sono stati creati numerosi modelli di business, che vengono costantemente modificati ed estesi da innumerevoli liberi professionisti e aziende in tutto il mondo in modo molto creativo, così da garantire la continua esistenza dell'intero movimento FLOSS.

Esercizi Guidati

1. Quali sono — in poche parole — le "quattro libertà" definite da Richard Stallman e dalla Free Software Foundation?

libertà 0	
libertà 1	
libertà 2	
libertà 3	

2. Che cosa significa la sigla FLOSS?

3. Hai sviluppato un software libero e vuoi assicurarti che tale software, ma anche tutte le opere future basate su di esso, rimangano liberi. Quale licenza scegli?

CC BY	
GPL versione 3	
Licenza BSD a 2 clausule	
LGPL	

4. Quali delle seguenti licenze definiresti permissive e quali copyleft?

Licenza BSD Semplificata	
GPL versione 3	
CC BY	
CC BY-SA	

5. Hai scritto un'applicazione web e l'hai pubblicata con una licenza libera. Come puoi guadagnare soldi con il tuo prodotto? Indica tre possibilità.

Esercizi Esplorativi

1. Sotto quale licenza (inclusa la versione) sono disponibili le seguenti applicazioni?

Apache HTTP Server	
MySQL Community Server	
Articoli di Wikipedia	
Mozilla Firefox	
GIMP	

2. Vuoi rilasciare il tuo software sotto GNU GPL v3. Quali passi dovresti seguire?

3. Hai scritto un software proprietario e vorresti combinarlo con del software libero sotto licenza GPL versione 3. Sei autorizzato a farlo, o che cosa devi prendere in considerazione?

4. Perché la Free Software Foundation ha rilasciato la *GNU Affero General Public License* (GNU AGPL) come supplemento alla GNU GPL?

5. Indica tre esempi di software libero offerti anche come “Business Edition”, per esempio in una versione a pagamento.

Sommario

In questa lezione hai imparato:

- Similitudini e differenze tra software libero e open source (FLOSS);
- Licenze FLOSS, loro importanza e problematiche relative;
- Licenze Copyleft vs. Licenze "Permissive";
- Modelli di business FLOSS.

Risposte agli Esercizi Guidati

1. Quali sono — in poche parole — le “quattro libertà” definite da Richard Stallman e dalla Free Software Foundation?

libertà 0	eseguire il software
libertà 1	studiare e modificare il software (codice sorgente)
libertà 2	distribuire il software
libertà 3	distribuire il software modificato

2. Che cosa significa la sigla FLOSS?

   Free/Libre Open Source Software

3. Hai sviluppato un software libero e vuoi assicurarti che tale software, ma anche tutte le opere future basate su di esso, rimangano liberi. Quale licenza scegli?

CC BY	
GPL versione 3	X
Licenza BSD a 2 clausule	
LGPL	

4. Quali delle seguenti licenze definiresti permissive e quali copyleft?

Licenza BSD Semplificata	permissiva
GPL versione 3	copyleft
CC BY	permissiva
CC BY-SA	copyleft

5. Hai scritto un’applicazione web e l’hai pubblicata con una licenza libera. Come puoi guadagnare soldi con il tuo prodotto? Indica tre possibilità.
   - Doppia licenza, per esempio offrendo una “Business Edition” a pagamento;
   - Offerta di hosting, servizio e supporto;
   - Sviluppo di estensioni proprietarie per i clienti.

Risposte agli Esercizi Esplorativi

1. Sotto quale licenza (inclusa la versione) sono disponibili le seguenti applicazioni?

Apache HTTP Server	Apache License 2.0
MySQL Community Server	GPL 2
Articoli di Wikipedia (Inglese)	Creative Commons Attribution Share-Alike (CC-BY-SA)
Mozilla Firefox	Mozilla Public License 2.0
GIMP	GPL 3

2. Vuoi rilasciare il tuo software sotto GNU GPL v3. Quali passi dovresti seguire?
   - Se necessario, proteggiti dal datore di lavoro, per esempio con una rinuncia al copyright in modo da poter indicare esattamente la licenza;
   - Aggiungi una nota di copyright a ogni file;
   - Aggiungi al tuo software un file chiamato `COPYING` con il testo completo della licenza;
   - Aggiungi un riferimento alla licenza in ogni file.
3. Hai scritto un software proprietario e vorresti combinarlo con del software libero sotto licenza GPL versione 3. Sei autorizzato a farlo, o che cosa devi prendere in considerazione?

   Le FAQ della Free Software Foundation forniscono queste informazioni: fintanto che il software proprietario e il software libero rimangono separati l'uno dall'altro, la combinazione è possibile. Tuttavia, devi assicurarti che questa separazione sia tecnicamente garantita e riconoscibile per gli utenti. Se si integra il software libero in modo che diventi parte integrante del prodotto, è anche necessario pubblicare il prodotto sotto GPL secondo il principio del copyleft.

4. Perché la Free Software Foundation ha rilasciato la *GNU Affero General Public License* (GNU AGPL) come supplemento alla GNU GPL?

   La GNU AGPL colma una lacuna di licenza che si presenta soprattutto con il software libero ospitato su un server: se uno sviluppatore apporta modifiche al software, non è obbligato ai sensi della GPL a rendere accessibili queste modifiche, poiché consente l'accesso al programma, ma non "ridistribuisce" il programma nel senso della GPL. La GNU AGPL, d'altra parte, stabilisce che il software deve essere reso disponibile per il download con tutte le modifiche.

5. Indica tre esempi di software libero offerti anche come “Business Edition”, per esempio in una versione a pagamento.

   MySQL, Zammad, Nextcloud.

1.4 Competenze ICT - Lavorare con Linux

Obiettivi LPI di riferimento

Linux Essentials version 1.6, Exam 010, Objective 1.4

Peso

2

Aree di Conoscenza Chiave

- Competenze sui desktop
- Uso della command line
- Utilizzi di Linux nell'industria, cloud computing e virtualizzazione

Di seguito è riportato un elenco parziale dei file, dei termini e dei comandi utilizzati

- Utilizzare un browser, problemi di privacy, opzioni di configurazione, ricerca sul Web e salvataggio di contenuti
- Terminale e console
- Problemi con le password
- Problemi e strumenti di privacy
- Uso di comuni applicazioni open source in presentazioni e progetti

1.4 Lezione 1

Certificazione:	Linux Essentials
Versione:	1.6
Argomento:	1 La Comunità Linux e una Carriera nell'Open Source
Obiettivo:	1.4 Competenze ICT - Lavorare con Linux
Lezione:	1 di 1

Introduzione

C'è stato un tempo in cui lavorare con Linux in ambiente desktop era considerato difficile poiché il sistema non disponeva di molte delle applicazioni desktop più sofisticate e degli strumenti di configurazione che gli altri sistemi operativi avevano. Del resto Linux era molto più recente di molti altri sistemi operativi. Detto questo, è stato più semplice iniziare con lo sviluppo delle applicazioni a Command Line indispensabili e lasciare gli strumenti grafici più complessi per una fase successiva. All'inizio, dal momento che Linux era innanzitutto destinato agli utenti più avanzati, non avrebbe dovuto essere un problema. Ma quei giorni sono oramai lontani. Oggi gli ambienti desktop Linux sono molto maturi e non lasciano nulla a desiderare in termini di funzionalità e semplicità di utilizzo. Tuttavia, la Command Line è ancora considerata un potente strumento utilizzato quotidianamente dagli utenti avanzati. In questa lezione esamineremo alcune delle competenze desktop di base di cui avrai bisogno per scegliere lo strumento migliore per il lavoro da svolgere, incluso l'accesso alla Command Line.

Interfacce Utente in Linux

Quando si utilizza un sistema Linux, si interagisce con una Command Line o con un'interfaccia utente grafica. Entrambi i modi permettono l'accesso a numerose applicazioni che consentono di svolgere quasi tutte le attività con il computer. Sebbene l'obiettivo 1.2 abbia già introdotto una serie di applicazioni di uso comune, inizieremo questa lezione con uno sguardo più da vicino agli ambienti desktop, ai modi per accedere al terminale e agli strumenti utilizzati per la presentazione e la gestione dei progetti.

Ambienti Desktop

Linux ha un approccio modulare in cui diverse parti del sistema sono sviluppate da diversi progetti e sviluppatori; ognuna delle parti soddisfa una specifica esigenza o obiettivo. Per questo motivo, ci sono diverse opzioni di ambienti desktop tra cui scegliere e, insieme ai gestori di pacchetti, l'ambiente desktop predefinito è una delle principali differenze tra le molte distribuzioni disponibili. A differenza dei sistemi operativi proprietari come Windows e macOS, in cui gli utenti sono limitati all'ambiente desktop fornito con il loro OS, c'è la possibilità di installare più ambienti e scegliere quello più adatto a te e alle tue esigenze.

Fondamentalmente, ci sono due principali ambienti desktop nel mondo Linux: *Gnome* e *KDE*. Sono entrambi molto completi, con una grande comunità alle spalle e con le medesime finalità, ma con approcci leggermente diversi. In poche parole, Gnome cerca di seguire il principio KISS ("keep it simple stupid"), con applicazioni molto snelle e pulite. Al contrario, KDE ha un'altra prospettiva con una più ampia selezione di applicazioni e offre all'utente l'opportunità di cambiare ogni impostazione di configurazione nell'ambiente.

Mentre le applicazioni Gnome sono basate sul toolkit GTK (scritto in linguaggio C), le applicazioni KDE fanno uso della libreria Qt (scritta in C ++). Uno degli aspetti più pratici per sviluppare applicazioni con lo stesso toolkit grafico è che le applicazioni tenderanno a condividere un aspetto simile, il che a sua volta dà all'utente un'impressione di uniformità. Un'altra caratteristica importante è che l'uso di una stessa libreria grafica condivisa per molte applicazioni usate di frequente può far risparmiare un po' di spazio di memoria e allo stesso tempo velocizzare i tempi di caricamento dopo che la libreria è stata caricata per la prima volta.

Accedere alla Command Line

Una delle applicazioni più importanti è l'emulatore grafico di terminale. Sono chiamati emulatori di terminale perché emulano realmente, in un ambiente grafico, i terminali seriali vecchio stile (spesso macchine telescriventi) che erano in realtà client collegati a una macchina remota su cui effettivamente avveniva l'elaborazione. Quelle macchine erano computer davvero semplici, senza alcuna interfaccia grafica, e venivano usati nelle prime versioni di Unix.

In Gnome, tale applicazione si chiama *Gnome Terminal*, mentre in KDE può essere presente come *Konsole*. Ma sono disponibili molte altre alternative, come *Xterm*. Queste applicazioni ci consentono di avere accesso a un ambiente a Command Line per poter interagire con una shell.

Dovresti quindi guardare il menu delle applicazioni della tua distribuzione preferita per trovare un'applicazione terminale. A parte qualche differenza tra loro, tutte le applicazioni ti offriranno ciò di cui hai bisogno per acquisire sicurezza nell'uso della Command Line.

Un altro modo per accedere al terminale è utilizzare il TTY virtuale. Puoi accedervi premendo `Ctrl` + `Alt` + `F#`. Considera `F#` come uno dei tasti funzione da 1 a 7. Probabilmente, alcune delle combinazioni iniziali potrebbero eseguire il gestore della sessione o il tuo ambiente grafico. Le altre mostreranno un prompt che richiede il tuo nome utente come nell'esempio qui sotto:

```
Ubuntu 18.10 arrelia tty3
arrelia login:
```

`arrelia` in questo caso è il nome host della macchina e `tty3` è il terminale disponibile dopo aver usato la combinazione di tasti sopra, più il tasto `F3`, ovvero `Ctrl` + `Alt` + `F3`.

Dopo aver inserito il nome utente e la password entrerai in un terminale, ma qui non ci sarà un ambiente grafico e quindi non sarai in grado di utilizzare il mouse o eseguire applicazioni grafiche senza prima avviare una sessione di X o di Wayland. Tuttavia questo va al di là dello scopo di questa lezione.

Presentazioni e Progetti

Lo strumento più importante per le presentazioni su Linux è *LibreOffice Impress*. Fa parte della suite per ufficio open source chiamata *LibreOffice*. Pensa a LibreOffice come un programma sostitutivo open source per l'equivalente *Microsoft Office*. Può anche aprire e salvare i file PPT e PPTX nativi di *Powerpoint*. Ma nonostante ciò, ti consiglio vivamente di utilizzare il formato di Impress nativo ODP. ODP fa parte del più ampio *Open Document Format*, che è uno standard internazionale per questo tipo di file. Ciò è particolarmente importante se desideri mantenere i tuoi documenti accessibili per molti anni e non vuoi preoccuparti dei problemi di compatibilità. Poiché si tratta di uno standard aperto, chiunque può implementare tale formato senza pagare royalties o licenze. Ciò ti consente anche di provare altri software di presentazione che potrebbero piacerti di più e di avere i tuoi file sempre con te, poiché è molto probabile che siano compatibili con i software più recenti.

Ma se preferisci il codice alle interfacce grafiche, ci sono alcuni strumenti tra cui scegliere. *Beamer* è una classe *LaTeX* che può creare presentazioni di diapositive da codice LaTeX. Lo stesso LaTeX è un sistema di composizione tipografico molto utilizzato per la scrittura di documenti scientifici

accademici, specialmente per la sua capacità di gestire simboli matematici complessi, che altri software hanno difficoltà a gestire. Se sei all'università e hai a che fare con equazioni e altre questioni matematiche, Beamer può farti risparmiare molto tempo.

L'altra opzione è *Reveal.js*, un fantastico pacchetto NPM (NPM è il gestore di pacchetti NodeJS predefinito) che ti consente di creare bellissime presentazioni utilizzando il web. Quindi, se sai usare HTML e CSS, Reveal.js ti fornirà la maggior parte del JavaScript necessario per creare presentazioni carine e interattive che si adatteranno bene a qualsiasi risoluzione e dimensione dello schermo.

Infine, se desideri un programma sostitutivo per *Microsoft Project*, puoi provare *GanttProject* o *ProjectLibre*. Entrambi sono molto simili alla controparte proprietaria e compatibili con i file di Project.

Usi Industriali di Linux

Linux è ampiamente utilizzato nei settori software e Internet. Siti come W3Techs riportano che circa il 68% dei server di siti web su Internet sono basati su Unix e la maggior parte di questi su Linux.

Questa ampiezza di utilizzo è dovuta non solo alla natura libera di Linux (sia nel senso di *birra gratis* sia in quello di *libertà di espressione*) ma anche alla sua stabilità, flessibilità e per le sue prestazioni. Queste caratteristiche consentono ai fornitori di offrire servizi con un costo inferiore e una maggiore scalabilità. Una parte significativa dei sistemi Linux viene oggi eseguita in cloud, su modelli IaaS (Infrastructure as a service), PaaS (Platform as a Service) o SaaS (Software as a Service).

IaaS è un modo per condividere le risorse di un server di grandi dimensioni offrendo l'accesso a macchine virtuali che sono, di fatto, più sistemi operativi in esecuzione come ospiti su una macchina host, su un importante "pezzo" di software chiamato *hypervisor*. L'hypervisor consente l'esecuzione di questi SO ospiti separando e gestendo le risorse disponibili sulla macchina host per tali SO ospiti. Questo è ciò che chiamiamo *virtualizzazione*. Nel modello IaaS, paghi solo per la frazione delle risorse utilizzate dalla tua infrastruttura.

Linux ha tre ben noti hypervisor open source: *Xen*, *KVM* e *VirtualBox*. Xen è probabilmente il più vecchio. KVM ha superato Xen come il più importante Hypervisor Linux. Il suo sviluppo è sponsorizzato da RedHat ed è utilizzato anche da altre realtà del settore, sia nei servizi cloud pubblici sia nelle configurazioni cloud private. VirtualBox appartiene a Oracle sin dalla sua acquisizione di Sun Microsystems ed è solitamente utilizzato dagli utenti finali per la sua facilità d'uso e di amministrazione.

PaaS e SaaS, d'altra parte, si basano sul modello IaaS, sia tecnicamente sia concettualmente. In PaaS, al posto di una macchina virtuale, gli utenti hanno accesso a una piattaforma dove possono distribuire ed eseguire le applicazioni. In questo caso, l'obiettivo è quello di ridurre gli sforzi richiesti per le attività di amministrazione di sistema e per gli aggiornamenti dei sistemi operativi. Heroku è un esempio comune di PaaS in cui il codice del programma può essere semplicemente eseguito senza preoccuparsi dei container e delle macchine virtuali sottostanti.

Infine, SaaS è il modello in cui di solito si paga un abbonamento per utilizzare semplicemente un software senza preoccuparsi di nient'altro. *Dropbox* e *Salesforce* sono due buoni esempi di SaaS. Alla maggior parte di questi servizi si accede tramite un browser web.

Un progetto come *OpenStack* è una raccolta di software open source che può utilizzare diversi hypervisor e altri strumenti per offrire un ambiente cloud IaaS completo *in loco*, sfruttando la potenza del cluster di computer nel proprio datacenter. Tuttavia, la configurazione di tale infrastruttura non è banale.

Problemi di Privacy nell'utilizzo di Internet

Il browser web è oggigiorno un software fondamentale su qualsiasi desktop, ma alcune persone non hanno ancora le conoscenze necessarie per utilizzarlo in modo sicuro. Sebbene sempre più servizi siano accessibili tramite un browser web, quasi tutte le azioni eseguite tramite un browser vengono tracciate e analizzate da vari attori. Garantire l'accesso ai servizi Internet e impedire il tracciamento è un aspetto importante dell'utilizzo di Internet in modo sicuro.

Tracciamento dei Cookie

Supponiamo che tu abbia navigato in un sito web di e-commerce, selezionando un prodotto che desideri e che tu lo abbia inserito nel carrello. Ma all'ultimo secondo hai deciso di ripensarci e di riflettere se ne avevi davvero bisogno. Dopo un po', inizi a vedere gli annunci dello stesso prodotto che ti "seguono" in giro per il web. Quando clicchi sugli annunci, vieni immediatamente reindirizzato alla pagina del prodotto di quel sito. Non è raro che i prodotti che hai inserito nel carrello siano ancora lì, in attesa che tu decida di dar loro un'occhiata. Ti sei mai chiesto come succeda? Come ti mostrino l'annuncio "giusto" su un'altra pagina web? La risposta a queste domande si chiama *tracciamento dei cookie*.

I cookie sono piccoli file che un sito web può salvare sul tuo computer al fine di memorizzare e recuperare alcuni tipi di informazioni utili alla tua navigazione. Sono in uso da molti anni e sono uno dei metodi più datati per memorizzare i dati lato client. Un buon esempio del loro utilizzo risiede negli ID univoci delle carte d'acquisto, in modo che, se torni sullo stesso sito web entro pochi giorni, il negozio potrà ricordarti i prodotti che hai inserito nel carrello durante la tua ultima visita e farti risparmiare tempo per ritrovarli.

Questo generalmente è un bene, poiché il sito web ti offre una funzionalità utile e non condivide alcun dato con terze parti. Ma per quanto riguarda gli annunci che ti vengono mostrati mentre navighi su altre pagine web? È qui che entrano in gioco le reti pubblicitarie. Le reti pubblicitarie sono aziende che offrono da un lato annunci pubblicitari per siti di e-commerce, come quello del nostro esempio, e monetizzazione per i siti web dall'altro. I creatori di contenuti come i blogger, per esempio, possono mettere a disposizione un po' di spazio per queste reti pubblicitarie sul loro blog, in cambio di una commissione sulle vendite generate dagli annunci pubblicitari.

Ma come fanno a sapere quale prodotto mostrarti? Di solito salvando un cookie dalla rete pubblicitaria nel momento in cui visiti o cerchi un determinato prodotto sul sito web di e-commerce. In questo modo la rete è in grado di recuperare le informazioni su quel cookie ovunque la rete abbia annunci pubblicitari, effettuando la correlazione con i prodotti a cui eri interessato. Questo è di solito uno dei modi più comuni per tracciare qualcuno su Internet. L'esempio che abbiamo mostrato sopra fa uso di un sito web di e-commerce per rendere le cose più tangibili, ma le piattaforme di social media fanno lo stesso con i loro pulsanti "Mi piace" o "Condividi" e con il login alla loro piattaforma.

Un modo per evitare questo è non consentire a siti web di terze parti di memorizzare cookie nel browser. In questo modo, solo il sito web che visiti può memorizzare i propri cookie. Ma tieni presente che alcune funzionalità "legittime" potrebbero non operare bene se lo fai, perché molti siti oggi si affidano a servizi di terze parti per funzionare. Pertanto, puoi cercare un gestore di cookie nei componenti aggiuntivi del tuo browser per avere un controllo dettagliato di quali cookie vengono memorizzati sul tuo computer.

Do Not Track (DNT)

Un altro malinteso comune riguarda una particolare impostazione del browser meglio nota come DNT. È l'acronimo di "Do Not Track" e può essere attivata praticamente su qualsiasi browser. Come nel caso della modalità privata, non è difficile trovare persone che credono di non essere tracciate se hanno attiva questa impostazione. Purtroppo non è sempre vero. Attualmente, DNT è solo un modo per dire ai siti web che visiti che non vuoi che ti traccino. Ma, in effetti, sono loro che decideranno se rispettare o meno la tua scelta. In altre parole, DNT è un modo per disattivare il tracciamento dei siti web, ma non vi è alcuna garanzia che questa richiesta venga rispettata.

Tecnicamente parlando questo viene fatto semplicemente inviando un flag aggiuntivo nell'header del protocollo di richiesta HTTP (`DNT: 1`) quando si richiedono i dati da un server web. Se vuoi saperne di più su questo argomento, il sito web https://allaboutdnt.com è un buon punto di partenza.

Finestre “Private”

Probabilmente hai notato le virgolette nel titolo qui sopra. Questo perché queste finestre non sono private come la maggior parte delle persone pensa. I nomi possono variare, ma possono essere chiamate “modalità privata”, scheda di “navigazione in incognito” o di "navigazione anonima", a seconda del browser che stai utilizzando.

In Firefox, puoi facilmente usarne una premendo i tasti `Ctrl` + `Shift` + `P`. In Chrome, premi semplicemente `Ctrl` + `Shift` + `N`. Ciò che in realtà il browser fa è aprire una nuova sessione, che normalmente non condivide alcuna configurazione o alcun dato dal tuo profilo standard. Alla chiusura della finestra privata, il browser cancellerà automaticamente tutti i dati generati da quella sessione, senza lasciare traccia sul computer utilizzato. Ciò significa che nessun dato personale, come cronologia, password o cookie viene memorizzato su quel computer.

Pertanto molte persone fraintendono questo concetto credendo di poter navigare in modo anonimo su Internet, il che non è completamente vero. Una cosa che fa la modalità privacy o la modalità di navigazione in incognito è evitare ciò che chiamiamo tracciamento dei cookie. Quando visiti un sito web, questo può memorizzare un piccolo file sul tuo computer contenente un ID che può essere utilizzato per tracciarti. A meno che tu non configuri il tuo browser in modo da non accettare cookie di terze parti, le reti pubblicitarie o altre società possono archiviare e recuperare quell’ID e di fatto tracciare la tua navigazione attraverso i siti web. Ma poiché i cookie memorizzati in una sessione in modalità privata vengono eliminati subito dopo la chiusura di quella sessione, tali informazioni andranno perse per sempre.

Oltre a questo, i siti web e gli altri *peers* in Internet possono però utilizzare molte altre tecniche per tracciarti. Quindi la modalità privata ti offre un certo livello di anonimato, ma è completamente privata solo sul computer che stai utilizzando. Se accedi al tuo account di posta elettronica o al tuo sito web bancario da un computer pubblico, come in un aeroporto o in un hotel, dovresti assolutamente utilizzare la modalità privata del browser. In altre situazioni potrebbero esserci dei vantaggi, ma è necessario capire esattamente quali rischi stai evitando e quali non hanno alcun impatto. Ogni volta che utilizzi un computer accessibile al pubblico, tieni presente che potrebbero esistere altre minacce alla sicurezza come malware o key logger. Fai attenzione ogni volta che inserisci informazioni personali, inclusi nomi utente e password, su tali computer o quando scarichi o copi dati riservati.

Scegliere la Giusta Password

Una delle situazioni più difficili che un utente deve affrontare è la scelta di una password sicura per i servizi che utilizza. Hai sicuramente sentito prima d’ora che non dovresti utilizzare combinazioni comuni come `qwerty`,`123456` o `654321`, né numeri facilmente indovinabili come il tuo compleanno (o quello di un parente) o il codice postale. Il motivo è che queste sono tutte

combinazioni molto prevedibili e saranno i primi tentativi di un utente malintenzionato per accedere al tuo account.

Esistono tecniche note per creare una password sicura. Una delle più famose è inventare una frase che ti ricordi quel servizio e scegliere le prime lettere di ogni parola. Supponiamo per esempio che io voglia creare una buona password per Facebook. In questo caso, potrei pensare a una frase del tipo "Io sarei felice se avessi 1000 amici come Mike". Prendi la prima lettera di ogni parola e la password finale sarà `Isfsa1000acM`. Ciò si tradurrebbe in una password di 12 caratteri, abbastanza lunga da essere difficile da indovinare e, allo stesso tempo, facile da ricordare (a patto che riesca a ricordare la frase e l'"algoritmo" per recuperare la password).

Le frasi di solito sono più facili da ricordare delle password, ma anche questo metodo ha i suoi limiti. Oggigiorno dobbiamo creare delle password per molteplici servizi e, poiché le usiamo con frequenze diverse, alla fine diventa molto difficile ricordare tutte le frasi nel momento in cui ne abbiamo bisogno. Quindi cosa possiamo fare? Potresti rispondere che la cosa più saggia da fare in questo caso è creare un paio di buone password e riutilizzarle per servizi simili, giusto?

Sfortunatamente, anche questa non è una buona idea. Probabilmente hai anche sentito che non dovresti riutilizzare la stessa password per servizi diversi. Il problema nel fare una cosa del genere è che un servizio specifico potrebbe far trapelare la tua password (sì, succede di continuo) e chiunque abbia accesso a essa proverà a utilizzare la stessa combinazione di email e password per altri servizi comuni in Internet nella speranza che tu abbia fatto esattamente questo: usare password riciclate. E indovina cosa succede? Nel caso in cui abbiano ragione, finirai per avere un problema non solo su un servizio ma su molti di essi. E credimi, tendiamo a pensare che a noi non succederà finché non è troppo tardi.

Quindi, cosa possiamo fare per proteggerci? Uno degli approcci più sicuri, oggi disponibili, è l'utilizzo di quello che viene chiamato *gestore di password*. I gestori di password sono dei software che in sostanza memorizzano tutte le tue password e i nomi utente in un formato crittato che può essere decrittato da una password principale. In questo modo devi solo ricordare una buona password poiché il gestore terrà tutte le altre al sicuro per te.

KeePass è uno dei gestori di password open source più famosi e ricchi di funzionalità tra quelli disponibili: memorizza le tue password in un file crittato all'interno del tuo filesystem. Il fatto che sia open source è una cosa importante per questo tipo di software, poiché garantisce che non ci sarà alcun uso dei tuoi dati perché qualsiasi sviluppatore può controllare il codice e sapere esattamente come funziona. Ciò fornisce un livello di trasparenza impossibile da raggiungere con del codice proprietario. KeePass ha versioni per la maggior parte dei sistemi operativi, inclusi Windows, Linux e macOS, così come per quelli mobili come iOS e Android. Include anche un sistema di plugin che è in grado di estendere le sue funzionalità ben oltre quelle standard.

Bitwarden è un'altra soluzione open source che ha un approccio simile, ma invece di archiviare i dati in un file, utilizzerà un server cloud. In questo modo, è più facile mantenere sincronizzati tutti i tuoi dispositivi e accedere in modo semplice alle tue password tramite il web. *Bitwarden* è uno dei pochi progetti che rende disponibile non solo i client, ma anche il server cloud, come software open source. Ciò significa che puoi ospitare la tua versione di Bitwarden e renderla disponibile a chiunque, come la tua famiglia o i dipendenti della tua azienda. Questo ti dà flessibilità ma anche pieno controllo su come le loro password vengono memorizzate e utilizzate.

Una delle cose più importanti da tenere a mente quando si utilizza un gestore di password è creare una password casuale per ogni diverso servizio poiché non sarà comunque necessario ricordarle. Sarebbe inutile utilizzare un gestore di password per archiviare password riciclate o facilmente individuabili. Pertanto, la maggior parte di essi ti offrirà un generatore di password casuali che potrai utilizzare per creare le password al posto tuo.

Crittografia

Ogni volta che i dati vengono trasmessi o archiviati è necessario prendere precauzioni per garantire che terze parti non possano accedere a essi. I dati trasmessi in Internet passano da una serie di router e reti in cui terze parti potrebbero avere accesso al traffico di rete. Allo stesso modo, i dati archiviati su supporti fisici potrebbero essere letti da chiunque ne entri in possesso. Per evitare questo tipo di accesso, le informazioni riservate dovrebbero essere crittografate prima di lasciare un dispositivo informatico.

TLS

Transport Layer Security (TLS) è un protocollo in grado di offrire sicurezza alle connessioni di rete facendo uso della crittografia. TLS è il successore di *Secure Sockets Layer* (SSL) che è stato deprecato a causa di gravi difetti. TLS si è anche evoluto un paio di volte per adattarsi e diventare più sicuro, quindi la sua versione attuale è la 1.3. Garantisce sia privacy sia autenticità facendo uso di quella che viene chiamata crittografia simmetrica e a chiave pubblica. Ciò significa che, una volta in uso, puoi essere certo che nessuno sarà in grado di intercettare o alterare la tua comunicazione con quel server durante quella sessione.

L'insegnamento più importante qui è riconoscere che un sito web sia affidabile. Dovresti cercare il simbolo del "lucchetto" nella barra degli indirizzi del browser. Se lo desideri, puoi fare clic su di esso per ispezionare il certificato, che riveste un ruolo fondamentale nel protocollo HTTPS.

TLS è ciò che viene utilizzato nel protocollo HTTPS (*HTTP su TLS*) per rendere possibile l'invio di dati sensibili (come il numero della carta di credito) attraverso il web. Spiegare come funziona TLS va ben oltre lo scopo di questa lezione, ma puoi trovare ulteriori informazioni su Wikipedia e su Mozilla wiki.

Crittografia di File ed E-mail con GnuPG

Esistono molti strumenti per proteggere i messaggi di posta elettronica, ma uno dei più importanti è sicuramente *GnuPG*. GnuPG sta per *GNU Privacy Guard* ed è un'implementazione open source di *OpenPGP*, che è uno standard internazionale codificato all'interno della RFC 4880.

GnuPG può essere utilizzato per firmare, crittare e decrittare testi, e-mail, file, directory e persino intere partizioni del disco. Funziona con la crittografia a chiave pubblica ed è molto utilizzato. In poche parole GnuPG crea una coppia di file che contengono la tua chiave pubblica e la tua chiave privata. Come suggerisce il nome, la chiave pubblica può essere a disposizione di tutti mentre la chiave privata deve essere tenuta segreta. Le persone utilizzeranno la tua chiave pubblica per crittare i dati che solo la tua chiave privata sarà in grado di decrittare.

Inoltre puoi utilizzare la tua chiave privata per firmare qualsiasi file o e-mail che possono essere validati dalla corrispondente chiave pubblica. Questa firma digitale funziona in modo analogo alla firma del mondo reale. Finché sei l'unico a possedere la tua chiave privata, il destinatario può essere sicuro che sei tu effettivamente l'autore della firma. Facendo uso della funzione crittografica di hash, GnuPG garantirà inoltre che non siano state apportate modifiche dopo la firma perché qualsiasi modifica al contenuto invaliderebbe la firma.

GnuPG è uno strumento molto potente e, per certi versi, anche complesso. Puoi trovare maggiori informazioni sul sito ufficiale e su Archlinux wiki (Archlinux wiki è un'ottima fonte di informazioni, anche se non usi Archlinux).

Crittografia del Disco

Un buon modo per proteggere i tuoi dati è crittografare l'intero disco o una partizione di esso. Esistono molti software open source che possono essere utilizzati per tale scopo. Anche il modo in cui funzionano e il livello di crittografia che offrono varia notevolmente. Sono disponibili due metodi di base: crittografia del dispositivo *in pila* (*stacked device encryption*) e *a blocchi* (*block device encryption*).

Le soluzioni di crittografia dei filesystem di tipo stacked vengono implementate sopra il filesystem esistente. Quando si utilizza questo metodo, i file e le directory vengono crittati prima di essere memorizzati nel filesystem e decrittati dopo essere stati letti. Ciò significa che i file sono archiviati sul filesystem host in una forma crittografata (il che significa che i loro contenuti, e solitamente anche i loro nomi di file/cartelle, sono sostituiti da dati apparentemente casuali), ma a parte questo, esistono ancora su quel filesystem nello stesso modo in cui esisterebbero senza crittografia, come file normali, collegamenti simbolici, hard link, etc.

Invece, la crittografia del dispositivo a blocchi avviene sotto il livello del filesystem, assicurandosi che tutto ciò che viene scritto su un dispositivo a blocchi venga crittografato. Se guardi il blocco

mentre è offline, apparirà come una grande massa di dati casuali e non sarai nemmeno in grado di dire quale tipo di filesystem sia senza prima decrittarlo. Ciò significa che non puoi dire cosa sia un file o una directory, quanto è grande e che tipo di dati sono, perché anche i metadati, la struttura delle directory e le autorizzazioni sono crittati.

Entrambi i metodi hanno i loro pro e contro. Tra tutte le alternative disponibili, dovresti dare un'occhiata a *dm-crypt*, che è di fatto lo standard per la crittografia a blocchi per i sistemi Linux, poiché è nativo del kernel. Può essere utilizzato con l'estensione *LUKS* (*Linux Unified Key Setup*), che è una specifica che implementa uno standard indipendente dalla piattaforma da utilizzare con diversi strumenti.

Se vuoi provare un metodo di crittografia stacked, dovresti dare un'occhiata a *EncFS*, che è probabilmente il modo più semplice per proteggere i dati su Linux poichè non richiede i privilegi di root e può essere eseguito su un filesystem esistente senza modifiche.

Infine, se hai bisogno di accedere ai dati su più piattaforme, dai un'occhiata a Veracrypt. È il successore di Truecrypt e consente la creazione di file, anche multimediali, crittografati che possono essere utilizzati su Linux, macOS e Windows.

Esercizi Guidati

1. Dovresti utilizzare una "finestra di navigazione in incognito" nel tuo browser nel caso in cui tu voglia:

Navigare in modo completamente anonimo in Internet	
Non lasciare traccia sul computer che stai utilizzando	
Attivare TLS per evitare il tracciamento dei cookie	
Utilizzare DNT	
Utilizzare la crittografia durante la trasmissione dei dati	

2. Cos'è OpenStack?

Un progetto che consente la creazione di IaaS privati	
Un progetto che consente la creazione di PaaS	
Un progetto che consente la creazione di SaaS	
Un hypervisor	
Un gestore di password open source	

3. Quali delle seguenti opzioni riportano validi software per la crittografia del disco?

RevealJS, EncFS e dm-crypt	
dm-crypt e KeePass	
EncFS e Bitwarden	
EncFS e dm-crypt	
TLS e dm-crypt	

4. Indica se le seguenti affermazioni sulla crittografia dei dispositivi con dm-crypt sono vere o false:

I file vengono crittografati prima di essere scritti sul disco	
L’intero filesystem è un blob crittografato	
Vengono crittografati solo i file e le directory, non i collegamenti simbolici	
Non richiede l’accesso come root	
È una crittografia del dispositivo a blocchi	

5. Beamer è:

Un meccanismo di crittografia	
Un hypervisor	
Un software di virtualizzazione	
Un componente di OpenStack	
Uno strumento di presentazione LaTeX	

Esercizi Esplorativi

1. La maggior parte delle distribuzioni viene fornita con Firefox installato di default (in caso contrario, dovrai prima installarlo). Installeremo ora un'estensione per Firefox chiamata *Lightbeam*. Puoi farlo premendo `Ctrl` + `Shift` + `A` e cercando "Lightbeam" nel campo di ricerca che verrà mostrato nella scheda aperta, oppure visitando la pagina delle estensioni con Firefox e cliccando sul pulsante "Installa": https://addons.mozilla.org/en-GB/firefox/addon/lightbeam-3-0/. Avvia quindi l'estensione facendo clic sulla sua icona e inizia a visitare alcune pagine web su altre schede per vedere cosa succede.
2. Qual è la cosa più importante quando si utilizza un gestore di password?

3. Usa il tuo browser web per navigare su https://haveibeenpwned.com/. Scopri a cosa serve il sito web e controlla se il tuo indirizzo di posta elettronica è stato affetto da qualche fuga di dati.

Sommario

Il terminale è un potente mezzo per interagire con il sistema ed esistono molti strumenti utili e ben collaudati da utilizzare in questo tipo di ambiente. Puoi accedere al terminale cercandone uno grafico nel menu dell'ambiente desktop o premendo `Ctrl` + `Alt` + `F#`.

Linux è ampiamente utilizzato nel settore tecnologico per offrire servizi IaaS, PaaS e SaaS. Ci sono tre hypervisor principali che giocano un ruolo importante nel supporto di questi servizi: Xen, KVM e Virtualbox.

Il browser è un software essenziale nell'informatica al giorno d'oggi, ma è necessario capire alcune cose per utilizzarlo in modo sicuro. DNT è solo un modo per dire al sito web che non vuoi essere tracciato, ma non esiste alcuna garanzia che venga rispettato. Le finestre di navigazione in incognito sono private solamente per il computer che stai utilizzando, ma proprio per questo motivo possono consentirti di evitare il tracciamento dei cookie.

TLS è in grado di crittografare la tua comunicazione in Internet, ma devi essere in grado di riconoscere quando è in uso. Anche l'utilizzo di password complesse è molto importante per mantenere la sicurezza, quindi l'idea migliore è delegare tale responsabilità a un gestore di password e consentire al software di creare password casuali per ogni sito a cui si accede.

Un altro modo per proteggere la tua comunicazione è firmare e crittografare i tuoi file, le tue cartelle e le tue email con GnuPG. dm-cypt ed EncFS sono due alternative per crittografare interi dischi o partizioni che utilizzano rispettivamente metodi di crittografia a blocchi e a stack.

Infine, LibreOffice Impress è un'alternativa open source e molto completa a Microsoft Powerpoint, ma esistono anche Beamer e RevealJS se preferisci creare presentazioni usando il codice invece delle GUI. ProjectLibre e GanttProject possono essere la scelta giusta se hai bisogno di una alternativa a Microsoft Project.

Risposte agli Esercizi Guidati

1. Dovresti utilizzare una "finestra di navigazione in incognito" nel tuo browser nel caso in cui tu voglia:

Navigare in modo completamente anonimo in Internet	
Non lasciare traccia sul computer che stai utilizzando	X
Attivare TLS per evitare il tracciamento dei cookie	
Utilizzare DNT	
Utilizzare la crittografia durante la trasmissione dei dati	

2. Cos'è OpenStack?

Un progetto che consente la creazione di IaaS privati	X
Un progetto che consente la creazione di PaaS	
Un progetto che consente la creazione di SaaS	
Un hypervisor	
Un gestore di password open source	

3. Quali delle seguenti opzioni riportano validi software per la crittografia del disco?

RevealJS, EncFS e dm-crypt	
dm-crypt e KeePass	
EncFS e Bitwarden	
EncFS e dm-crypt	X
TLS e dm-crypt	

4. Indica se le seguenti affermazioni sulla crittografia dei dispositivi con dm-crypt sono vere o false:

I file vengono crittografati prima di essere scritti sul disco	V
L’intero filesystem è un blob crittografato	V
Vengono crittografati solo i file e le directory, non i collegamenti simbolici	F
Non richiede l’accesso come root	F
È una crittografia del dispositivo a blocchi	V

5. Beamer è:

Un meccanismo di crittografia	
Un hypervisor	
Un software di virtualizzazione	
Un componente di OpenStack	
Uno strumento di presentazione LaTeX	X

Risposte agli Esercizi Esplorativi

1. La maggior parte delle distribuzioni viene fornita con Firefox installato di default (in caso contrario, dovrai prima installarlo). Installeremo ora un'estensione per Firefox chiamata *Lightbeam*. Puoi farlo premendo `Ctrl` + `Shift` + `A` e cercando "Lightbeam" nel campo di ricerca che verrà mostrato nella scheda aperta, oppure visitando la pagina delle estensioni con Firefox e cliccando sul pulsante "Installa": https://addons.mozilla.org/en-GB/firefox/addon/lightbeam-3-0/. Avvia quindi l'estensione facendo clic sulla sua icona e inizia a visitare alcune pagine web su altre schede per vedere cosa succede.

   Ricordi quei cookie che abbiamo detto che possono condividere i tuoi dati con diversi servizi quando visiti un sito web? Questo è esattamente ciò che ti mostrerà questa estensione. Lightbeam è un esperimento di Mozilla che cerca di rivelare i siti principali e di terze parti con cui interagisci visitando un singolo URL. Questo contenuto non è di solito visibile all'utente medio e può mostrare che a volte un singolo sito web è in grado di interagire con una dozzina, o più, di servizi.

2. Qual è la cosa più importante quando si utilizza un gestore di password?

   Quando si utilizza un gestore di password, la cosa più importante da tenere a mente è memorizzare la password principale e utilizzare una password casuale univoca per ognuno dei diversi servizi.

3. Usa il tuo browser web per navigare su https://haveibeenpwned.com/. Scopri a cosa serve il sito web e controlla se il tuo indirizzo di posta elettronica è stato affetto da qualche fuga di dati.

   Il sito web mantiene un database di informazioni di login le cui password sono state interessate da una fuga di password. Consente la ricerca di un indirizzo di posta elettronica e mostra se tale indirizzo di posta elettronica è stato incluso in un database pubblico di credenziali rubate. È probabile che anche il tuo indirizzo di posta elettronica sia stato interessato da una di queste fughe di password. In tal caso, assicurati di aver aggiornato le tue password di recente. Se non utilizzi già un gestore di password, dai un'occhiata a quelli consigliati in questa lezione.

Linux Professional Institute

Argomento 2: Trovare il Proprio Modo di Operare su un Sistema Linux

2.1 Nozioni di Base sulla Command Line

Obiettivi LPI di riferimento

Linux Essentials version 1.6, Exam 010, Objective 2.1

Peso

3

Aree di Conoscenza Chiave

- Shell di base
- Sintassi della riga di comando
- Variabili
- Uso delle virgolette

Di seguito è riportato un elenco parziale dei file, dei termini e dei comandi utilizzati

- Bash
- `echo`
- `history`
- La variabile d'ambiente `PATH`
- `export`
- `type`

2.1 Lezione 1

Certificazione:	Linux Essentials
Versione:	1.6
Argomento:	2 Trovare il Proprio Modo di Operare su un Sistema Linux
Obiettivo:	2.1 Nozioni di Base sulla Command Line
Lezione:	1 di 2

Introduzione

Le moderne distribuzioni Linux offrono una vasta gamma di interfacce utente grafiche, ma un amministratore avrà sempre bisogno di sapere come lavorare con la Command Line, chiamata anche *shell*. La shell è un programma che consente la comunicazione di tipo testuale tra il sistema operativo e l'utente. Di solito è un programma testuale che legge l'input dell'utente e lo interpreta sotto forma di comandi per il sistema.

Esistono diverse shell su Linux; questi sono solo alcuni esempi:

- Bourne-again shell (Bash)
- C shell (csh o tcsh, la versione migliorata di csh)
- Korn shell (ksh)
- Z shell (zsh)

Su Linux la più comune è la *Shell Bash*. È anche quella che verrà utilizzata nei nostri esempi ed esercizi.

Quando si utilizza una shell interattiva, l'utente immette i comandi nel cosiddetto prompt. Per ogni distribuzione Linux, il prompt predefinito può avere un aspetto leggermente diverso, ma di solito segue la seguente struttura:

```
username@hostname directory_corrente tipo_di_shell
```

Su Ubuntu o Debian GNU/Linux, il prompt per un utente ordinario sarà probabilmente simile a questo:

```
carol@mycomputer:~$
```

Il prompt del superuser sarà simile a questo:

```
root@mycomputer:~#
```

Su CentOS o Red Hat Linux, il prompt per un utente ordinario sarà invece simile a questo:

```
[dave@mycomputer ~]$
```

E il prompt del superuser sarà simile a questo:

```
[root@mycomputer ~]#
```

Spieghiamo ora ogni componente della struttura:

`username`
: Il nome dell'utente che esegue la shell.

`hostname`
: Il nome dell'host su cui viene eseguita la shell. Esiste anche un comando `hostname` con il quale puoi visualizzare o impostare il nome host del sistema.

`directory_corrente`
: La directory in cui si trova attualmente la shell. `~` indica che la shell è nella directory home dell'utente corrente.

`tipo_di_shell`
: `$` indica che la shell è eseguita da un utente ordinario.

`#` indica che la shell è eseguita dal superutente `root`.

Poiché non abbiamo bisogno di privilegi speciali, nei successivi esempi utilizzeremo un prompt non privilegiato. Per brevità, useremo semplicemente `$` per indicare il prompt.

Struttura della Command Line

La maggior parte dei comandi inseriti sulla Command Line segue la medesima struttura di base:

```
comando  [opzione(i)/parametro(i)...]  [argomento(i)...]
```

Prendi il seguente comando come esempio:

```
$ ls -l /home
```

Spieghiamo il senso di ogni componente:

Comando

Il programma che l'utente intende eseguire: `ls` nell'esempio sopra riportato.

Opzione(i)/Parametro(i)

Un "modificatore" che cambia in qualche modo il comportamento del comando, come `-l` nell'esempio sopra riportato. È possibile accedere alle opzioni in forma breve e lunga. Per esempio, `-l` è identico a `--format=long`.

È anche possibile combinare più opzioni e, per la forma abbreviata, le lettere di solito possono essere inserite assieme. Per esempio, i seguenti comandi fanno tutti la stessa cosa:

```
$ ls -al
$ ls -a -l
$ ls --all --format=long
```

Argomento(i)

I dati aggiuntivi richiesti dal programma, per esempio un nome di file o un percorso, come `/home` nell'esempio sopra riportato.

L'unica parte obbligatoria di questa struttura è il comando stesso. In generale, tutti gli altri elementi sono opzionali, ma un programma potrebbe richiedere che vengano specificati determinati parametri, opzioni o argomenti.

NOTE La maggior parte dei comandi mostra una breve panoramica dei comandi disponibili quando vengono eseguiti con il parametro `--help`. Presto esamineremo altri modi per saperne di più sui comandi Linux.

Tipi di Comportamento dei Comandi

La shell supporta due tipi di comandi:

Interni

Questi comandi fanno parte della shell stessa e non sono programmi separati. Esistono circa 30 comandi di questo tipo. Il loro scopo principale è eseguire attività all'interno della shell (per esempio `cd`, `set`, `export`).

Esterni

Questi comandi risiedono in singoli file. Questi file sono in genere programmi o script binari. Quando viene eseguito un comando che non è incorporato nella shell (*shell bultin*), la shell utilizza la variabile `PATH` per cercare un file eseguibile con lo stesso nome del comando. Oltre ai programmi installati con il gestore di pacchetti della distribuzione, gli utenti possono anche creare i propri comandi esterni.

Il comando `type` mostra a quale tipologia appartiene uno specifico comando:

```
$ type echo
echo is a shell builtin
$ type man
man is /usr/bin/man
```

Quoting

Come utente Linux, dovrai creare o manipolare file o variabili in vari modi. Questo è facile quando si lavora con nomi di file brevi e con semplici valori, ma diventa più complicato quando, per esempio, sono coinvolti spazi, caratteri speciali e variabili. Le shell mettono a disposizione una funzionalità chiamata *quoting* che incapsula tali dati utilizzando vari tipi di virgolette (" ", ' '). In Bash, esistono tre tipi di virgolette:

- Virgolette doppie
- Virgolette singole
- Caratteri di escape

Per esempio, i seguenti comandi non si comportano allo stesso modo a causa del quoting:

```
$ TWOWORDS="two words"
$ touch $TWOWORDS
$ ls -l
-rw-r--r-- 1 carol carol     0 Mar 10 14:56 two
-rw-r--r-- 1 carol carol     0 Mar 10 14:56 words
$ touch "$TWOWORDS"
$ ls -l
-rw-r--r-- 1 carol carol     0 Mar 10 14:56  two
-rw-r--r-- 1 carol carol     0 Mar 10 14:58 'two words'
-rw-r--r-- 1 carol carol     0 Mar 10 14:56  words
$ touch '$TWOWORDS'
$ ls -l
-rw-r--r-- 1 carol carol     0 Mar 10 15:00 '$TWOWORDS'
-rw-r--r-- 1 carol carol     0 Mar 10 14:56  two
-rw-r--r-- 1 carol carol     0 Mar 10 14:58 'two words'
-rw-r--r-- 1 carol carol     0 Mar 10 14:56  words
```

NOTE La riga con `TWOWORDS=` indica una variabile Bash che noi stessi abbiamo creato. Introdurremo le variabili in seguito. Questo esempio ha solo lo scopo di mostrarti come il quoting influisce sull'output delle variabili.

Virgolette Doppie

Le virgolette doppie indicano alla shell di trattare il testo tra virgolette ("...") come caratteri normali. Tutti i caratteri speciali perdono il loro significato, eccetto `$` (dollaro), `\` (*backslash*) e `` ` `` (*backquote*). Ciò significa che è ancora possibile utilizzare variabili, sostituzione di comandi e funzioni aritmetiche.

Per esempio, la sostituzione della variabile `$USER` non è influenzata dalle virgolette doppie:

```
$ echo I am $USER
I am tom
$ echo "I am $USER"
I am tom
```

Il carattere "spazio", invece, perde il suo significato di separatore di argomenti:

```
$ touch new file
$ ls -l
```

```
-rw-rw-r-- 1 tom students 0 Oct 8 15:18 file
-rw-rw-r-- 1 tom students 0 Oct 8 15:18 new
$ touch "new file"
$ ls -l
-rw-rw-r-- 1 tom students 0 Oct 8 15:19 new file
```

Come puoi vedere, nel primo esempio il comando `touch` crea due singoli file; il comando interpreta infatti le due stringhe come singoli argomenti. Nel secondo esempio, il comando interpreta entrambe le stringhe come un unico argomento e quindi crea solo un file. Tuttavia è buona norma evitare il carattere spazio nei nomi dei file. Potrebbe essere utilizzato, come alternativa, un trattino basso (`_`) o un punto (`.`).

Virgolette Singole

Le virgolette singole non hanno le eccezioni delle virgolette doppie. Revocano qualsiasi significato speciale attribuito a ogni carattere. Prendiamo uno dei primi esempi tra i precedenti sopra riportati:

```
$ echo I am $USER
I am tom
```

Se utilizzi le virgolette singole, vedrai un risultato diverso:

```
$ echo 'I am $USER'
I am $USER
```

Il comando ora visualizza la stringa esatta senza sostituire la variabile.

Caratteri di Escape

Possiamo utilizzare i *caratteri di escape* per rimuovere i significati speciali dei caratteri in Bash. Tornando alla variabile d'ambiente `$USER`:

```
$ echo $USER
carol
```

Notiamo che, per impostazione predefinita, il contenuto della variabile viene visualizzato nel terminale. Tuttavia, se dovessimo precedere il segno del dollaro con il carattere backslash (`\`), il significato speciale del segno del dollaro verrebbe annullato. Ciò a sua volta non permetterà a

Bash di espandere il valore della variabile inserendo il nome utente della persona che esegue il comando, ma invece interpreterà il nome della variabile in modo letterale:

```
$ echo \$USER
$USER
```

Se ricordi, possiamo ottenere risultati simili a questo utilizzando le virgolette singole, che stampano in modo letterale a video il contenuto di ciò racchiudono. Tuttavia, il carattere di escape funziona in modo diverso, dicendo a Bash di ignorare qualsiasi significato speciale che possa avere il carattere che lo segue.

Esercizi Guidati

1. Suddividi le seguenti righe nelle singole componenti: comando, opzione(i)/parametro(i) ed argomento(i):
   - Esempio: `cat -n /etc/passwd`

Comando:	`cat`
Opzione:	`-n`
Argomento:	`/etc/passwd`

   - `ls -l /etc`

Comando:	
Opzione:	
Argomento:	

   - `ls -l -a`

Comando	
Opzione:	
Argomento:	

   - `cd /home/user`

Comando:	
Opzione:	
Argomento:	

2. Scopri la tipologia dei seguenti comandi:

   Esempio:

`pwd`	Shell builtin
`mv`	Comando esterno

`cd`	

`cat`	
`exit`	

3. Risolvi i seguenti comandi che utilizzano le virgolette:

   Esempio:

`echo "$HOME is my home directory"`	`echo /home/user is my home directory`

`touch "$USER"`	
`touch 'touch'`	

Esercizi Esplorativi

1. Con un unico comando e utilizzando l'espansione delle parentesi graffe in Bash (controlla la pagina di manuale di Bash), crea 5 file numerati da 1 a 5 con il prefisso `game` (`game1`, `game2`, ...).

2. Elimina tutti i 5 file che hai appena creato con un solo comando, utilizzando un carattere speciale diverso dal precedente (cerca *Pathname Expansion* nelle pagine di manuale di Bash).

3. Esistono altri modi per far interagire due comandi tra loro? Quali sono?

Sommario

In questa lezione hai imparato:

- I principi della shell Linux;
- Cos'è la shell Bash;
- La struttura della Command Line;
- Una introduzione al quoting.

Comandi utilizzati negli esercizi:

`bash`
La shell più popolare sui computer Linux.

`echo`
Scrive del testo sul terminale.

`ls`
Elenca il contenuto di una directory.

`type`
Mostra come viene eseguito uno specifico comando.

`touch`
Crea un file vuoto o aggiorna la data di modifica di un file esistente.

`hostname`
Mostra o modifica il nome host di un sistema.

Risposte agli Esercizi Guidati

1. Suddividi le seguenti righe nelle singole componenti: comando, opzione(i)/parametro(i) ed argomento(i):
   - `ls -l /etc`

Comando:	`ls`
Opzione:	`-l`
Argomento:	`/etc`

   - `ls -l -a`

Comando:	`ls`
Opzione:	`-l -a`
Argomento:	

   - `cd /home/user`

Comando:	`cd`
Opzione:	
Argomento:	`/home/user`

2. Scopri la tipologia dei seguenti comandi:

`cd`	Shell builtin
`cat`	Comando esterno
`exit`	Shell builtin

3. Risolvi i seguenti comandi che utilizzano le virgolette:

`touch "$USER"`	`tom`
`touch 'touch'`	Crea un file chiamato `touch`

Risposte agli Esercizi Esplorativi

1. Con un unico comando e utilizzando l'espansione delle parentesi graffe in Bash (controlla la pagina di manuale di Bash), crea 5 file numerati da 1 a 5 con il prefisso `game` (`game1`, `game2`, ...).

   È possibile utilizzare gli intervalli per indicare i numeri da 1 a 5 all'interno di un unico comando:

   ```
   $ touch game{1..5}
   $ ls
   game1  game2  game3  game4  game5
   ```

2. Elimina tutti i 5 file che hai appena creato con un solo comando, utilizzando un carattere speciale diverso dal precedente (cerca *Pathname Expansion* nelle pagine di manuale di Bash).

   Poiché tutti i file iniziano con `game` e terminano con un singolo carattere (un numero da 1 a 5 in questo caso), può essere utilizzato `?` come carattere speciale per indicare l'ultimo carattere nel nome del file:

   ```
   $ rm game?
   ```

3. Esistono altri modi per far interagire due comandi tra loro? Quali sono?

   Sì, un comando potrebbe, per esempio, scrivere i dati in un file che viene poi elaborato da un altro comando. Linux può anche prendere l'output di un comando e utilizzarlo come input per un altro comando. Questo meccanismo si chiama *piping*; ne scopriremo di più in una delle prossime lezioni.

2.1 Lezione 2

Certificazione:	Linux Essentials
Versione:	1.6
Argomento:	2 Trovare il Proprio Modo di Operare su un Sistema Linux
Obiettivo:	2.1 Nozioni di Base sulla Command Line
Lezione:	2 di 2

Introduzione

Tutte le shell gestiscono una serie di informazioni di stato durante ogni sessione di shell. Queste informazioni di runtime possono cambiare durante la sessione e influenzano il comportamento della shell stessa. Questi dati vengono utilizzati anche dai programmi per individuare elementi della configurazione del sistema. La maggior parte di questi dati è archiviata nelle così dette *variabili*, che tratteremo in questa lezione.

Variabili

Le variabili sono contenitori di dati, come testo o numeri. Una volta impostatolo, è possibile accedere al valore di una variabile in un secondo momento. Le variabili hanno un nome che consente di accedere a una specifica variabile anche quando il suo contenuto cambia. Sono uno strumento molto comune nella maggior parte dei linguaggi di programmazione.

Nella maggior parte delle shell Linux esistono due tipi di variabili:

Variabili locali

Queste variabili sono disponibili solamente per il processo shell corrente. Se crei una variabile locale e poi avvii un altro programma da questa shell, quel programma non potrà più accedere alla variabile. Poiché non vengono ereditate dai sottoprocessi, queste variabili sono chiamate *variabili locali*.

Variabili ambientali

Queste variabili sono disponibili sia in una sessione di shell specifica sia nei sottoprocessi generati da questa sessione di shell. Queste variabili possono essere utilizzate per passare le informazioni di configurazione ai comandi che vengono eseguiti. Poiché i programmi possono accedere a queste variabili, vengono chiamate *variabili ambientali*. La maggior parte delle variabili ambientali si trova in lettere maiuscole (per esempio `PATH`, `DATE`, `USER`). Un insieme di variabili d'ambiente standard fornisce, per esempio, informazioni sulla directory home dell'utente o sul tipo di terminale. Talvolta il set completo di tutte le variabili ambientali viene chiamato *ambiente*.

NOTE Le variabili non sono permanenti. Quando la shell in cui sono state impostate viene chiusa, tutte le variabili e il loro contenuto vengono persi. La maggior parte delle shell fornisce dei file di configurazione che contengono variabili che vengono impostate ogni volta che viene avviata una nuova shell. Le variabili che devono essere impostate in modo permanente devono essere aggiunte a uno di questi file di configurazione.

Gestire le Variabili

In qualità di amministratore di sistema dovrai creare, modificare o rimuovere sia le variabili locali sia quelle ambientali.

Lavorare con le Variabili Locali

È possibile impostare una variabile locale utilizzando l'operatore `=` (uguale). Un semplice assegnamento creerà una variabile locale:

```
$ greeting=hello
```

NOTE Non lasciare alcuno spazio prima o dopo l'operatore `=`.

Puoi mostrare qualsiasi variabile utilizzando il comando `echo`. Il comando di solito visualizza il testo nella sezione degli argomenti:

```
$ echo greeting
greeting
```

Per accedere al valore della variabile dovrai utilizzare `$` (simbolo del dollaro) davanti al nome della variabile.

```
$ echo $greeting
hello
```

Come si può vedere, la variabile è stata creata. Ora apri un'altra shell e prova a visualizzare il contenuto della variabile creata.

```
$ echo $greeting
```

Non viene visualizzato nulla. Ciò dimostra che le variabili esistono sempre solamente in una specifica shell.

Per verificare che la variabile sia effettivamente una variabile locale, prova a generare un nuovo processo e controlla se questo processo può accedere alla variabile. Possiamo farlo avviando un'altra shell e facendo eseguire a questa shell il comando `echo`. Poiché la nuova shell viene eseguita in un nuovo processo, non erediterà le variabili locali dal suo processo padre:

```
$ echo $greeting world
hello world
$ bash -c 'echo $greeting world'
world
```

NOTE Assicurati di utilizzare le virgolette singole nell'esempio qui sopra.

Per rimuovere una variabile, dovrai utilizzare il comando `unset`:

```
$ echo $greeting
hey
$ unset greeting
$ echo $greeting
```

NOTE `unset` richiede il nome della variabile come argomento. Pertanto non si può aggiungere `$` al nome, poiché ciò risolverebbe la variabile e passerebbe il valore

della variabile a `unset` al posto del nome della variabile.

Lavorare con le Variabili Globali

Per rendere una variabile disponibile ai sottoprocessi, possiamo trasformarla da variabile locale a variabile ambientale. Questo viene fatto tramite il comando `export`. Quando viene invocato con il nome della variabile, questa viene aggiunta all'ambiente della shell:

```
$ greeting=hello
$ export greeting
```

NOTE Di nuovo, assicurati di non usare `$` quando esegui il comando `export` perché vuoi passare il nome della variabile, non il suo contenuto.

Un modo più semplice per creare una variabile ambientale è combinare entrambi i comandi di cui sopra, assegnando il valore della variabile nella sezione del comando dedicata all'argomento.

```
$ export greeting=hey
```

Controlliamo di nuovo se la variabile è accessibile ai sottoprocessi:

```
$ export greeting=hey
$ echo $greeting world
hey world
$ bash -c 'echo $greeting world'
hey world
```

Un altro modo per utilizzare le variabili ambientali consiste nell'inserirle davanti ai comandi. Possiamo testarlo con la variabile ambientale `TZ` che contiene il fuso orario. Questa variabile è usata dal comando `date` per determinare quale fuso orario visualizzare:

```
$ TZ=EST date
Thu 31 Jan 10:07:35 EST 2019
$ TZ=GMT date
Thu 31 Jan 15:07:35 GMT 2019
```

Puoi visualizzare tutte le variabili ambientali usando il comando `env`.

La Variabile PATH

La variabile `PATH` è una delle più importanti variabili ambientali in un sistema Linux. Memorizza un elenco di directory, separate da due punti, che contengono programmi eseguibili che possono essere chiamati come comandi dalla shell Linux.

```
$ echo $PATH
/home/user/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
```

Per aggiungere una nuova directory alla variabile, sarà necessario utilizzare il segno di interpunzione dei due punti (`:`).

```
$ PATH=$PATH:new_directory
```

Ecco un esempio:

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
$ PATH=$PATH:/home/user/bin
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/home/user/bin
```

Come puoi vedere `$PATH` viene utilizzato nel nuovo valore assegnato a `PATH`. Questa variabile viene risolta durante l'esecuzione del comando e garantisce che il contenuto originale della variabile venga preservato. Ovviamente puoi usare anche altre variabili nell'operazione di assegnamento:

```
$ mybin=/opt/bin
$ PATH=$PATH:$mybin
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/home/user/bin:/opt/bin
```

La variabile `PATH` deve essere gestita con cautela, poiché è fondamentale per lavorare dalla Command Line. Consideriamo la seguente variabile `PATH`:

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Per scoprire come la shell invoca uno specifico comando, può essere eseguito `which` con il nome del comando come argomento. Possiamo, per esempio, tentare di scoprire dove è memorizzato `nano`:

```
$ which nano
/usr/bin/nano
```

Come si può vedere, l'eseguibile `nano` si trova nella directory `/usr/bin`. Rimuoviamo tale directory dalla variabile e controlliamo se il comando funziona ancora:

```
$ PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/sbin:/bin:/usr/games
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/sbin:/bin:/usr/games
```

Cerchiamo nuovamente il comando `nano`:

```
$ which nano
which: no nano in (/usr/local/sbin:/usr/local/bin:/usr/sbin:/sbin:/bin:/usr/games)
```

Come si può vedere, il comando non viene trovato e quindi non viene eseguito. Il messaggio di errore spiega anche il motivo per cui il comando non è stato trovato e dove è stato cercato.

Riaggiungiamo le directory e proviamo a eseguire nuovamente il comando.

```
$ PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
$ which nano
/usr/bin/nano
```

Ora il nostro comando funziona di nuovo.

TIP L'ordine degli elementi in `PATH` definisce anche l'ordine di ricerca. Verrà eseguita la prima corrispondenza eseguibile trovata durante la ricerca tra i percorsi.

Esercizi Guidati

1. Crea una variabile locale `number`.

2. Crea una variabile ambientale `ORDER` utilizzando uno dei due metodi sopra riportati.

3. Visualizza sia i nomi delle variabili sia il loro contenuto.

4. Quali sono gli ambiti delle variabili create in precedenza?

Esercizi Esplorativi

1. Crea una variabile locale `nr_files` e assegnale il numero di righe trovate nel file `/etc/passwd`. Suggerimento: controlla il comando `wc` e la sostituzione dei comandi senza dimenticare le virgolette.

2. Crea una variabile ambientale `ME`. Assegnale il valore della variabile `USER`.

3. Aggiungi il valore della variabile `HOME` a `ME`, utilizzando il delimitatore `:`. Visualizza il contenuto della variabile `ME`.

4. Utilizzando l'esempio della data sopra riportato, crea una variabile chiamata `today` e assegnale la data di uno dei fusi orari.

5. Crea un'altra variabile chiamata `today1` e assegnale la data del sistema.

Sommario

In questa lezione hai imparato:

- I tipi di variabili;
- Come creare le variabili;
- Come gestire le variabili.

Comandi utilizzati negli esercizi:

`env`
: Mostra l'ambiente corrente.

`echo`
: Genera del testo come output.

`export`
: Rende le variabili locali disponibili ai sottoprocessi.

`unset`
: Rimuove una variabile.

Risposte agli Esercizi Guidati

1. Crea una variabile locale `number`.

```
$ number=5
```

2. Crea una variabile ambientale `ORDER` utilizzando uno dei due metodi sopra riportati.

```
$ export ORDER=desc
```

3. Visualizza sia i nomi delle variabili sia il loro contenuto.

```
$ echo number
number
$ echo ORDER
ORDER
$ echo $number
5
$ echo $ORDER
desc
```

4. Quali sono gli ambiti delle variabili create in precedenza?
   - L'ambito della variabile locale `number` è solamente la shell corrente.
   - L'ambito della variabile d'ambiente `ORDER` è la shell corrente e tutte le sottoshell da essa generate.

Risposte agli Esercizi Esplorativi

1. Crea una variabile locale `nr_files` e assegnale il numero di righe trovate nel file `/etc/passwd`. Suggerimento: controlla il comando `wc` e la sostituzione dei comandi senza dimenticare le virgolette.

```
$ nr_files=`wc -l /etc/passwd`
```

2. Crea una variabile ambientale `ME`. Assegnale il valore della variabile `USER`.

```
$ export ME=$USER
```

3. Aggiungi il valore della variabile `HOME` a `ME`, utilizzando il delimitatore `:`. Visualizza il contenuto della variabile `ME`.

```
$ ME=$ME:$HOME
$ echo $ME
user:/home/user
```

4. Utilizzando l'esempio della data sopra riportato, crea una variabile chiamata `today` e assegnale la data di uno dei fusi orari.

   Nell'esempio seguente vengono utilizzati i fusi orari GMT ed EST, ma qualsiasi fuso orario è valido.

```
$ today=$(TZ=GMT date)
$ echo $today
Thu 31 Jan 15:07:35 GMT 2019
```

   o

```
$ today=$(TZ=EST date)
$ echo $today
Thu 31 Jan 10:07:35 EST 2019
```

5. Crea un'altra variabile chiamata `today1` e assegnale la data del sistema.

   Supponendo di essere nell'orario GMT:

```
$ today1=$(date)
$ echo $today1
Thu 31 Jan 10:07:35 EST 2019
```

2.2 Utilizzo della Command Line per Ottenere Aiuto

Obiettivi LPI di riferimento

Linux Essentials version 1.6, Exam 010, Objective 2.2

Peso

2

Aree di Conoscenza Chiave

- Le pagine man
- Le pagine info

Di seguito è riportato un elenco parziale dei file, dei termini e dei comandi utilizzati

- `man`
- `info`
- `/usr/share/doc/`
- `locate`

2.2 Lezione 1

Certificazione:	Linux Essentials
Versione:	1.6
Argomento:	2 Trovare il Proprio Modo di Operare su un Sistema Linux
Obiettivo:	2.2 Utilizzo della Command Line per Ottenere Aiuto
Lezione:	1 di 1

Introduzione

La Command Line è uno strumento molto complesso. Ogni comando ha le sue specifiche opzioni, per cui la documentazione è fondamentale quando si lavora con un sistema Linux. Oltre alla directory `/usr/share/doc/`, che contiene la maggior parte della documentazione, diversi altri strumenti forniscono informazioni sull'uso dei comandi Linux. Questo capitolo si concentra sui metodi per accedere a tale documentazione per ottenere aiuto.

Esistono molti metodi per ottenere aiuto dalla Command Line di Linux: `man`, `help` e `info` sono solo alcuni di questi. Per Linux Essentials, ci concentreremo su `man` e `info` poiché sono gli strumenti più frequentemente utilizzati per ottenere aiuto.

Un altro argomento di questo capitolo sarà la ricerca dei file. Lavorerai principalmente con il comando `locate`.

Ottenere Aiuto dalla Command Line

Aiuto Interno

Quando vengono eseguiti con il parametro `--help`, la maggior parte dei comandi mostra alcune brevi istruzioni sul loro utilizzo. Sebbene non tutti i comandi forniscano questa opzione è comunque una prima prova da fare per saperne di più sui parametri di un comando. Tieni presente che le indicazioni ottenute da `--help` sono spesso piuttosto brevi rispetto alle altre fonti di documentazione di cui parleremo nel resto di questa lezione.

Pagine Man

La maggior parte dei comandi prevede una pagina di manuale o una pagina "man". Questa documentazione viene solitamente installata con il software ed è possibile accedervi con il comando `man`. Il comando la cui pagina man deve essere visualizzata viene aggiunto a `man` come argomento:

```
$ man mkdir
```

Questo comando apre la pagina man di `mkdir`. È possibile utilizzare i tasti freccia su e giù o la barra spaziatrice per navigare nella pagina man. Per uscire dalla pagina man premi `Q`.

Ogni pagina man è suddivisa in un massimo di 11 sezioni, sebbene molte di queste siano opzionali:

Sezione	**Descrizione**
NAME	Nome del comando e breve descrizione
SYNOPSIS	Descrizione della sintassi del comando
DESCRIPTION	Descrizione degli effetti del comando
OPTIONS	Opzioni disponibili
ARGUMENTS	Argomenti disponibili
FILES	File ausiliari
EXAMPLES	Un esempio della Command Line
SEE ALSO	Riferimenti incrociati agli argomenti correlati
DIAGNOSTICS	Messaggi di avviso ed errore
COPYRIGHT	Autore(i) del comando

Sezione	**Descrizione**
BUGS	Eventuali limitazioni note del comando

In realtà, la maggior parte delle pagine man non contiene tutte queste parti.

Le pagine man sono organizzate in otto categorie, numerate da 1 a 8:

Categoria	**Descrizione**
1	Comandi utente
2	Chiamate di sistema
3	Funzioni della libreria C
4	Driver e file di dispositivo
5	File di configurazione e formati di file
6	Giochi
7	Miscellanea
8	Comandi dell'amministratore di sistema
9	Funzioni del kernel (non standard)

Ogni pagina man appartiene esattamente a una sezione. Tuttavia, più sezioni possono avere pagine man con lo stesso nome. Prendiamo come esempio il comando `passwd`. Questo comando può essere utilizzato per cambiare la password di un utente. Dato che `passwd` è un comando utente, la sua pagina man si trova nella sezione 1. Oltre al comando `passwd`, anche il file del database delle password `/etc/passwd` ha una pagina man che è chiamata anch'essa `passwd`. Poiché questo file è un file di configurazione, appartiene alla sezione 5. Quando si fa riferimento a una pagina man, la categoria viene spesso aggiunta al nome della pagina man, come in `passwd(1)` o `passwd(5)`, per identificare la rispettiva pagina man.

Per impostazione predefinita `man passwd` mostra la prima pagina man disponibile, in questo caso `passwd(1)`. La categoria della pagina man desiderata può essere specificata nel comando come, per esempio, `man 1 passwd` o `man 5 passwd`.

Abbiamo già discusso come navigare in una pagina man e come tornare alla Command Line. Internamente `man` utilizza il comando `less` per mostrare il contenuto della pagina man. `less` ti consente di cercare del testo all'interno di una pagina man. Per cercare la parola `linux` puoi semplicemente usare `/linux` per effettuare una ricerca in avanti rispetto al punto in cui ti trovi nella pagina, o `?Linux` per avviare una ricerca all'indietro. Questa azione evidenzia tutti i risultati trovati e sposta la pagina alla prima corrispondenza evidenziata. In entrambi i casi puoi premere

`N` per passare alla corrispondenza successiva. Per avere ulteriori informazioni su queste funzioni aggiuntive, premi `H` e verrà visualizzato un menu con tutte le informazioni.

Pagine Info

Un altro strumento che ti sarà d'aiuto mentre lavori con un sistema Linux è rappresentato dalle pagine info. Le pagine info sono solitamente più dettagliate delle pagine man e sono formattate come ipertesti, in modo simile alle pagine web in Internet.

Le pagine info possono essere visualizzate in questo modo:

```
$ info mkdir
```

Per ciascuna di queste pagine il comando `info` legge un file di informazioni che è strutturato in singoli nodi all'interno di un albero. Ogni nodo contiene un semplice argomento e il comando `info` include dei collegamenti ipertestuali che possono aiutarti a muoverti da uno all'altro. È possibile accedere al collegamento premendo invio con il cursore posizionato su uno degli asterischi iniziali.

Come `man`, lo strumento `info` prevede anche dei comandi di navigazione all'interno della pagina. Per saperne di più su questi comandi premi `?` all'interno della pagina info. Questi strumenti ti aiuteranno a navigare nella pagina più facilmente e a capire come accedere ai nodi e a muoverti all'interno dell'albero dei nodi.

La Directory `/usr/share/doc/`

Come accennato in precedenza, la directory `/usr/share/doc/` contiene la maggior parte della documentazione dei comandi utilizzati dal sistema. `/usr/share/doc/` contiene una directory per la maggior parte dei pacchetti installati sul sistema. Il nome della directory è solitamente il nome del pacchetto e talvolta include anche la sua versione. Queste directory contengono un file `README` o `readme.txt` con la documentazione di base del pacchetto. Oltre al file `README`, la cartella può contenere anche altri file di documentazione, come per esempio il registro delle modifiche (changelog) che contiene la cronologia del programma in dettaglio, o esempi di file di configurazione per lo specifico pacchetto.

Le informazioni all'interno del file `README` variano da un pacchetto all'altro. Tutti i file sono in formato testo e quindi possono essere letti con un qualunque editor di testo di tua scelta. Il numero esatto e il tipo di file dipendono dal pacchetto. Controlla alcune di queste directory per avere una panoramica del loro contenuto.

Trovare i File

Il Comando `locate`

Un sistema Linux è costituito da numerose directory e file. Linux ha molti strumenti per trovare un particolare file all'interno del sistema. Il più veloce è il comando `locate`.

`locate` cerca all'interno di un database e poi restituisce ogni nome che corrisponde alla stringa indicata:

```
$ locate note
/lib/udev/keymaps/zepto-znote
/usr/bin/zipnote
/usr/share/doc/initramfs-tools/maintainer-notes.html
/usr/share/man/man1/zipnote.1.gz
```

Il comando `locate` supporta anche l'uso dei caratteri jolly e delle espressioni regolari; quindi la stringa di ricerca non deve corrispondere al nome completo del file desiderato. Potrai scoprirne di più sulle espressioni regolari in uno dei prossimi capitoli.

Per impostazione predefinita `locate` si comporta come se il pattern di ricerca fosse tra asterischi; quindi `locate PATTERN` è uguale a `locate *PATTERN*`. Ciò ti consente di indicare solamente delle sottostringhe invece del nome esatto del file. Puoi modificare questo comportamento usando le diverse opzioni che trovi spiegate nella pagina man di `locate`.

Poiché `locate` legge da un database, potresti anche non trovare un file che hai creato di recente. Il database è gestito da un programma chiamato `updatedb`. Di solito viene eseguito periodicamente, ma se hai privilegi di root e hai bisogno che il database venga aggiornato immediatamente, puoi eseguire tu stesso il comando `updatedb` in qualsiasi momento.

Il Comando `find`

`find` è un altro strumento molto popolare utilizzato per cercare i file. Questo comando ha un approccio diverso rispetto al comando `locate`. Il comando `find` cerca in modo ricorsivo in un albero di directory, includendo le sue sottodirectory. `find` esegue una ricerca di questo tipo a ogni invocazione e non mantiene un database come `locate`. In analogia con `locate`, `find` supporta anche i caratteri jolly e le espressioni regolari.

`find` richiede almeno il percorso in cui deve cercare. Inoltre è possibile aggiungere le cosiddette espressioni per filtrare i file da visualizzare. Un esempio è l'espressione `-name`, che cerca i file con uno specifico nome:

```
~$ cd Downloads
~/Downloads
$ find . -name thesis.pdf
./thesis.pdf
~/Downloads
$ find ~ -name thesis.pdf
/home/carol/Downloads/thesis.pdf
```

Il primo comando `find` cerca il file nella directory corrente `Downloads`, mentre il secondo cerca il file nella directory home dell'utente.

Il comando `find` è molto complesso e quindi non sarà trattato nell'esame Linux Essentials. Tuttavia, è uno strumento potente e particolarmente utile nel quotidiano.

Esercizi Guidati

1. Usa il comando `man` per scoprire cosa fa ogni comando:

Comando	**Descrizione**
`ls`	Visualizza il contenuto di una directory.
`cat`	
`cut`	
`cd`	
`cp`	
`mv`	
`mkdir`	
`touch`	
`wc`	
`passwd`	
`rm`	
`rmdir`	
`more`	
`less`	
`whereis`	
`head`	
`tail`	
`sort`	
`tr`	
`chmod`	
`grep`	

2. Apri la pagina info di `ls` e identifica il MENU.
   - Quali opzioni hai?
   - Trova l'opzione che ti consente di ordinare l'output in base alla data di modifica.

3. Visualizza il percorso dei primi 3 file `README`. Usa il comando `man` per individuare l'opzione corretta di `locate`.

4. Crea un file chiamato `test` nella tua directory home. Trova il suo percorso assoluto con il comando `locate`.

5. L'hai trovato subito? Che cosa hai dovuto fare affinché `locate` lo trovasse?

6. Utilizzando il comando `find` cerca il file di test che hai creato in precedenza. Che sintassi hai usato e qual è il percorso assoluto?

Esercizi Esplorativi

1. C'è un comando nella tabella sopra riportata che non ha una pagina `man`. Qual è questo comando e perché pensi non abbia una pagina man?

2. Utilizzando i comandi nella tabella sopra riportata crea il seguente albero di file. I nomi che iniziano con una lettera maiuscola indicano delle Directory, mentre quelli che iniziano con una lettera minuscola indicano dei file.

```
User
├── Documents
|   ├──Hello
|   |   ├──hey2
|   |   ├──helloa
|   |   └──ola5
|   └──World
|       └──earth9
├── Downloads
|   ├──Music
|   └──Songs
|       ├──collection1
|       └──collection2
├── Test
|   └── passa
└── test
```

3. Visualizza sullo schermo la directory di lavoro corrente, includendo le sue sottodirectory.

4. Cerca all'interno dell'albero tutti i file che terminano con un numero.

5. Rimuovi tutti i rami della directory con un unico comando.

Sommario

In questa lezione hai imparato:

- Come ottenere aiuto;
- Come usare il comando `man`;
- Come navigare in una pagina `man`;
- Le diverse sezioni di una pagina `man`;
- Come usare il comando `info`;
- Come navigare tra i diversi nodi;
- Come cercare i file all'interno del sistema.

Comandi utilizzati negli esercizi:

`man`
Visualizza una pagina man.

`info`
Visualizza una pagina info.

`locate`
Cerca nel database di `locate` i file con un specifico nome.

`find`
Cerca nel file system i nomi che corrispondono a un insieme di criteri di selezione.

`updatedb`
Aggiorna il database di `locate`.

Risposte agli Esercizi Guidati

1. Usa il comando `man` per scoprire cosa fa ogni comando:

Comando	**Descrizione**
`ls`	Visualizza il contenuto di una directory
`cat`	Concatena o visualizza file di testo
`cut`	Rimuove sezioni da un file di testo
`cd`	Passa a un'altra directory
`cp`	Copia un file
`mv`	Sposta un file (può essere utilizzato anche per rinominare)
`mkdir`	Crea una nuova directory
`touch`	Crea un file o aggiorna la data e l'ora dell'ultima modifica di un file esistente
`wc`	Conta il numero di parole, di righe o di byte di un file
`passwd`	Modifica la password di un utente
`rm`	Cancella un file
`rmdir`	Cancella una directory
`more`	Visualizza i file di testo una schermata alla volta
`less`	Visualizza i file di testo, consentendo di scorrere verso l'alto o verso il basso di una riga o di una pagina alla volta
`whereis`	Visualizza il percorso di un programma specificato e dei relativi file di manuale
`head`	Visualizza le prime righe di un file
`tail`	Visualizza le ultime righe di un file
`sort`	Ordina un file numericamente o alfabeticamente
`tr`	Converte o elimina caratteri in un file

Comando	Descrizione
`chmod`	Modifica i diritti di accesso di un file
`grep`	Cerca all'interno di un file

2. Apri la pagina info di `ls` e identifica il MENU.
   - Quali opzioni hai?
     - Which files are listed (Quali file sono elencati)
     - What information is listed (Quali informazioni vengono visualizzate)
     - Sorting the output (Ordinare l'output)
     - Details about version sort (Dettagli sull'ordinamento per versione)
     - General output formatting (Formato generale dell'output)
     - Formatting file timestamps (Formato dei timestamp)
     - Formatting the file names (Formato dei nomi dei file)
   - Trova l'opzione che ti consente di ordinare l'output in base alla data di modifica.

     `-t` o `--sort=time`

3. Visualizza il percorso dei primi 3 file `README`. Usa il comando `man` per individuare l'opzione corretta di `locate`.

```
$ locate -l 3 README
/etc/alternatives/README
/etc/init.d/README
/etc/rc0.d/README
```

4. Crea un file chiamato `test` nella tua directory home. Trova il suo percorso assoluto con il comando `locate`.

```
$ touch test
$ locate test
/home/user/test
```

5. L'hai trovato subito? Che cosa hai dovuto fare affinché `locate` lo trovasse?

```
$ sudo updatedb
```

Il file è stato appena creato e quindi non è presente nel database.

6. Utilizzando il comando `find` cerca il file di test che hai creato in precedenza. Che sintassi hai usato e qual è il percorso assoluto?

```
$ find ~ -name test
```

o

```
$ find . -name test
/home/user/test
```

Risposte agli Esercizi Esplorativi

1. C'è un comando nella tabella sopra riportata che non ha una pagina `man`. Qual è questo comando e perché pensi non abbia una pagina man?

   Il comando `cd`. Non ha una pagina man poichè è un comando interno alla shell (built-in).

2. Utilizzando i comandi nella tabella sopra riportata crea il seguente albero di file. I nomi che iniziano con una lettera maiuscola indicano delle Directory, mentre quelli che iniziano con una lettera minuscola indicano dei file.

```
User
├── Documents
|   ├──Hello
|   |   ├──hey2
|   |   ├──helloa
|   |   └──ola5
|   └──World
|       └──earth9
├── Downloads
|   ├──Music
|   └──Songs
|       ├──collection1
|       └──collection2
├── Test
|   └── passa
└── test
```

   La soluzione è una combinazione dei comandi `mkdir` e `touch`.

3. Visualizza sullo schermo la directory di lavoro corrente, includendo le sue sottodirectory.

```
$ ls -R
```

4. Cerca all'interno dell'albero tutti i file che terminano con un numero.

```
$ find ~ -name "*[0-9]"
$ locate "*[0-9]"
```

5. Rimuovi tutti i rami della directory con un unico comando.

```
$ rm -r Documents Downloads Test test
```

2.3 Utilizzo di Directory e Elenchi di File

Obiettivi LPI di riferimento

Linux Essentials version 1.6, Exam 010, Objective 2.3

Peso

2

Aree di Conoscenza Chiave

- File, directory
- File and directory nascoste
- Directory home
- Percorsi assoluti e relativi

Di seguito è riportato un elenco parziale dei file, dei termini e dei comandi utilizzati

- Opzioni comuni di `ls`
- Listato recursivo
- `cd`
- `.` e `..`
- `home` e `~`

Linux Professional Institute

2.3 Lezione 1

Introduzione

Certificazione:	Linux Essentials
Versione:	1.6
Argomento:	2 Trovare il Proprio Modo di Operare su un Sistema Linux
Obiettivo:	2.3 Utilizzo di Directory e Elenchi di File
Lezione:	1 di 2

File e Directory

Il filesystem Linux è simile ai filesystem degli altri sistemi operativi in quanto contiene *file* e *directory*. I file contengono dati, per esempio testo leggibile dall'uomo, programmi eseguibili o dati binari utilizzati dal computer. Le directory vengono utilizzate per organizzare il filesystem; possono contenere file e altre directory.

```
$ tree

Documents
├── Mission-Statement.txt
└── Reports
    └── report2018.txt

1 directory, 2 files
```

In questo esempio, `Documents` è una directory che contiene un file (`Mission-Statement.txt`) e una *sottodirectory* (`Reports`). La directory `Reports` a sua volta contiene un file chiamato `report2018.txt`. La directory `Documents` è ritenuta *padre* della directory `Reports`.

TIP Se il comando `tree` non è disponibile sul tuo sistema, installalo usando il gestore di pacchetti della tua distribuzione Linux. Fai riferimento alla lezione sulla gestione dei pacchetti per imparare a farlo.

Nomi di File e Directory

I nomi di file e directory in Linux possono contenere lettere minuscole e maiuscole, numeri, spazi e caratteri speciali. Tuttavia, poiché molti caratteri speciali hanno un significato particolare nella shell Linux, è buona norma non utilizzare spazi o caratteri speciali quando si attribuiscono nomi a file o directory. Gli spazi, per esempio, richiedono che il *carattere di escape* `\` sia inserito correttamente:

```
$ cd Mission\ Statements
```

Fai inoltre riferimento al nome file `report2018.txt`. I nomi dei file possono contenere un *suffisso* che viene dopo il punto (`.`). A differenza che in Windows, questo suffisso non ha alcun significato speciale in Linux; è lì per una maggiore usabilità. Nel nostro esempio `.txt` ci indica che si tratta di un file di testo semplice, sebbene possa tecnicamente contenere qualsiasi tipo di dati.

Navigare nel Filesystem

Ottenere la Posizione Corrente

Poiché le shell Linux, come per esempio Bash, sono basate su testo è importante conoscere la posizione corrente durante la navigazione nel filesystem. Il *prompt dei comandi* fornisce questa informazione:

```
user@hostname ~/Documents/Reports $
```

Tieni presente che informazioni come `user` e `hostname` saranno trattate nelle future lezioni. Dal prompt ora sappiamo che la nostra posizione corrente è la directory `Reports`. Allo stesso modo, il comando `pwd` stamperà la directory di lavoro corrente (è l'acronimo di *print working directory*):

```
user@hostname ~/Documents/Reports $ pwd
```

```
/home/user/Documents/Reports
```

La relazione tra le directory è rappresentata da una barra obliqua (`/`). Sappiamo che `Reports` è una sottodirectory di `Documents`, che è una sottodirectory di `user`, che si trova in una directory chiamata `home`. `home` non sembra avere una directory padre, ma non è affatto vero. La directory padre di `home` si chiama *root*, ed è rappresentata dalla prima barra obliqua (`/`). Parleremo della directory root in una delle prossime lezioni.

Nota che l'output del comando `pwd` è leggermente diverso dal percorso mostrato sul prompt dei comandi. Invece di `/home/user`, il prompt dei comandi contiene una tilde (`~`). La tilde è un carattere speciale che rappresenta la directory home dell'utente. Ne discuteremo più in dettaglio nella prossima lezione.

Elencare il Contenuto di una Directory

Il contenuto della directory corrente viene elencato con il comando `ls`:

```
user@hostname ~/Documents/Reports $ ls
report2018.txt
```

Nota che `ls` non fornisce informazioni sulla directory padre. Allo stesso modo, per impostazione predefinita, `ls` non mostra alcuna informazione sul contenuto delle sottodirectory. `ls` può solo "vedere" ciò che si trova nella directory corrente.

Cambiare la Directory Corrente

La navigazione in Linux avviene principalmente tramite il comando `cd`, che è usato per cambiare directory (è l'acronimo di *change directory*). Usando il comando `pwd` visto in precedenza, notiamo che la nostra directory corrente è `/home/user/Documents/Reports`. Possiamo cambiare la nostra directory corrente inserendo un nuovo percorso:

```
user@hostname ~ $ cd /home/user/Documents
user@hostname ~/Documents $ pwd
/home/user/Documents
user@hostname ~/Documents $ ls
Mission-Statement.txt Reports
```

Dalla nostra nuova posizione, possiamo "vedere" `Mission-Statement.txt` e la nostra sottodirectory `Reports`, ma non il contenuto di questa nostra sottodirectory. Possiamo tornare indietro a `Reports` in questo modo:

```
user@hostname ~/Documents $ cd Reports
user@hostname ~/Documents/Reports $ pwd
/home/user/Documents/Reports
user@hostname ~/Documents/Reports $ ls
report2018.txt
```

Ora siamo tornati al punto di partenza.

Percorsi Assoluti e Relativi

Il comando `pwd` stampa sempre un *percorso assoluto*. Ciò significa che il percorso contiene ogni passaggio, dall'inizio del filesystem (`/`) alla fine (`Reports`). I percorsi assoluti iniziano sempre con `/`.

```
/
└── home
    └── user
        └── Documents
            └── Reports
```

Il percorso assoluto contiene tutte le informazioni richieste per arrivare a `Reports` da qualsiasi punto del filesystem. Lo svantaggio è che è noioso da digitare.

Il secondo esempio (`cd Reports`) è stato molto più semplice da digitare. Questo è un esempio di *percorso relativo*. I percorsi relativi sono più brevi, ma hanno significato solo in relazione alla posizione corrente. Considera questa analogia: vengo a trovarti a casa tua. Mi dici che il tuo amico vive nella porta accanto. Capirò quella posizione perché è relativa alla mia posizione attuale. Ma se me lo dici per telefono, non riuscirò a trovare la casa del tuo amico: dovrai darmi l'indirizzo completo.

Percorsi Relativi Speciali

La shell Linux ci offre dei modi per abbreviare i percorsi durante la navigazione. Per rivelare i primi percorsi speciali, inseriamo il comando `ls` con il flag `-a`. Questo flag modifica il comando `ls` in modo che vengano elencati *tutti* i file e le directory, inclusi i file e le directory nascoste:

```
user@hostname ~/Documents/Reports $ ls -a
.
..
```

```
report2018.txt
```

NOTE Puoi fare riferimento alla pagina `man` di `ls` per capire cosa fa `-a` in questo caso.

Questo comando ha rivelato due ulteriori risultati: questi sono percorsi speciali. Non rappresentano nuovi file o directory, ma piuttosto rappresentano delle directory che già conosci:

`.`
: Indica la *posizione corrente* (in questo caso `Reports`).

`..`
: Indica la *directory padre* (in questo caso `Documents`).

Di solito non è necessario utilizzare il percorso relativo speciale per la posizione corrente. È più facile e più comprensibile digitare `report2018.txt` che `./report2018.txt`. Ma ci sono usi per `.` che imparerai nelle prossime lezioni. Per il momento concentriamoci sul percorso relativo della directory padre:

```
user@hostname ~/Documents/Reports $ cd ..
user@hostname ~/Documents $ pwd
/home/user/Documents
```

L'esempio di `cd` è molto più semplice se si usa `..` invece del percorso assoluto. Inoltre, possiamo combinare questo pattern per navigare molto velocemente nell'albero dei file.

```
user@hostname ~/Documents $ cd ../..
$ pwd
/home
```

Esercizi Guidati

1. Per ciascuno dei seguenti percorsi, individua se è *assoluto* o *relativo*:

`/home/user/Downloads`	
`../Reports`	
`/var`	
`docs`	
`/`	

2. Osserva la seguente struttura di file. Nota: le directory terminano con una barra obliqua (`/`) quando `tree` viene invocato con l'opzione `-F`. Avrai bisogno di privilegi elevati per eseguire il comando `tree` nella directory root (`/`). Il seguente è un esempio di output e non è indicativo di una struttura di directory completa. Usalo per rispondere alle seguenti domande:

```
$ sudo tree -F /

/
├── etc/
│   ├── network/
│   │   └── interfaces
│   ├── systemd/
│   │   ├── resolved.conf
│   │   ├── system/
│   │   ├── system.conf
│   │   ├── user/
│   │   └── user.conf
│   └── udev/
│       ├── rules.d/
│       └── udev.conf
└── home/
    ├── lost+found/
    └── user/
        └── Documents/

12 directories, 5 files
```

   Usa questa struttura per rispondere alle seguenti domande.

   Un utente inserisce i seguenti comandi:

```
$ cd /etc/udev
$ ls -a
```

Quale sarà l'output del comando `ls -a`?

3. Inserisci il comando più breve possibile per ciascuna delle seguenti situazioni:
   - La tua posizione corrente è la directory root (`/`). Inserisci il comando per spostarti in `lost+found` all'interno della directory `home` (esempio):

     ```
     $ cd home/lost+found
     ```

   - La tua posizione corrente è la directory root (`/`). Inserisci il comando per spostarti nella directory chiamata `/etc/network/`:
   - La tua posizione corrente è la directory `/home/user/Documents/`. Spostati nella directory chiamata `/etc/`.
   - La tua posizione corrente è la directory `/etc/systemd/system/`. Spostati nella directory chiamata `/home/user/`.

4. Considera i seguenti comandi:

```
$ pwd
/etc/udev/rules.d
$ cd ../../systemd/user
$ cd ..
$ pwd
```

Qual è l'output dell'ultimo comando `pwd`?

Esercizi Esplorativi

1. Supponiamo che un utente abbia inserito i seguenti comandi:

   ```
   $ mkdir "this is a test"
   $ ls
   this is a test
   ```

   Quale comando `cd` ti permette di entrare in questa directory?

2. Prova di nuovo, ma, dopo aver digitato `cd this`, premi il tasto TAB. Che cosa viene visualizzato ora sul prompt?

   Questo è un esempio di *completamento automatico*, uno strumento preziosissimo non solo per risparmiare tempo, ma anche per prevenire errori di ortografia.

3. Prova a creare una directory il cui nome contenga un carattere `\`. Visualizza il nome della directory tramite `ls` ed elimina la directory.

Sommario

In questa lezione hai imparato:

- Le basi del filesystem Linux;
- La differenza tra directory *padre* e *sottodirectory*;
- La differenza tra percorsi di file *assoluti* e percorsi di file *relativi*;
- I percorsi relativi speciali `.` e `..`;
- Come navigare nel filesystem usando `cd`;
- Come mostrare la posizione corrente usando `pwd`;
- Come elencare *tutti* i file e le directory usando `ls -a`.

In questa lezione sono stati discussi i seguenti comandi:

`cd`
: Cambia la directory corrente.

`pwd`
: Stampa il percorso della directory di lavoro corrente.

`ls`
: Elenca il contenuto di una directory e mostra le proprietà dei file.

`mkdir`
: Crea una nuova directory.

`tree`
: Visualizza un elenco gerarchico di un albero di directory.

Risposte agli Esercizi Guidati

1. Per ciascuno dei seguenti percorsi, individua se è *assoluto* o *relativo*:

`/home/user/Downloads`	assoluto
`../Reports`	relativo
`/var`	assoluto
`docs`	relativo
`/`	assoluto

2. Osserva la seguente struttura di file. Nota: le directory terminano con una barra obliqua (`/`) quando `tree` viene invocato con l'opzione `-F`. Avrai bisogno di privilegi elevati per eseguire il comando `tree` nella directory root (`/`). Il seguente è un esempio di output e non è indicativo di una struttura di directory completa. Usalo per rispondere alle seguenti domande:

```
$ sudo tree -F /

/
├── etc/
│   ├── network/
│   │   └── interfaces
│   ├── systemd/
│   │   ├── resolved.conf
│   │   ├── system/
│   │   ├── system.conf
│   │   ├── user/
│   │   └── user.conf
│   └── udev/
│       ├── rules.d/
│       └── udev.conf
└── home/
    ├── lost+found/
    └── user/
        └── Documents/

12 directories, 5 files
```

   Un utente inserisce i seguenti comandi:

```
$ cd /etc/udev
$ ls -a
```

Quale sarà l'output del comando `ls -a`?

```
. .. rules.d udev.conf
```

3. Inserisci il comando più breve possibile per ciascuna delle seguenti situazioni:
   - La tua posizione corrente è la directory root (`/`). Inserisci il comando per spostarti in `lost+found` all'interno della directory `home` (esempio):

     ```
     $ cd home/lost+found
     ```

   - La tua posizione corrente è la directory root (`/`). Inserisci il comando per spostarti nella directory chiamata `/etc/network/`:

     ```
     $ cd etc/network
     ```

   - La tua posizione corrente è la directory `/home/user/Documents/`. Spostati nella directory chiamata `/etc/`.

     ```
     $ cd /etc
     ```

   - La tua posizione corrente è la directory `/etc/systemd/system/`. Spostati nella directory chiamata `/home/user/`.

     ```
     $ cd /home/user
     ```

4. Considera i seguenti comandi:

```
$ pwd
/etc/udev/rules.d
$ cd ../../systemd/user
$ cd ..
$ pwd
```

Qual è l'output dell'ultimo comando `pwd`?

```
/etc/systemd
```

Risposte agli Esercizi Esplorativi

1. Supponiamo che un utente abbia inserito i seguenti comandi:

   ```
   $ mkdir "this is a test"
   $ ls
   this is a test
   ```

   Quale comando `cd` ti permette di entrare in questa directory?

   ```
   $ cd this\ is\ a\ test
   ```

2. Prova di nuovo, ma, dopo aver digitato `cd this`, premi il tasto TAB. Che cosa viene visualizzato ora sul prompt?

   ```
   $ cd this\ is\ a\ test
   ```

   Questo è un esempio di *completamento automatico*, uno strumento preziosissimo non solo per risparmiare tempo, ma anche per prevenire errori di ortografia.

3. Prova a creare una directory il cui nome contenga un carattere `\`. Visualizza il nome della directory tramite `ls` ed elimina la directory.

   Puoi eseguire l'escape della barra rovesciata utilizzando un'altra barra rovesciata (`\\`) o utilizzare le virgolette singole o doppie attorno al nome completo della directory:

   ```
   $ mkdir my\\dir
   $ ls
   'my\dir'
   $ rmdir 'my\dir'
   ```

2.3 Lezione 2

Certificazione:	Linux Essentials
Versione:	1.6
Argomento:	2 Trovare il Proprio Modo di Operare su un Sistema Linux
Obiettivo:	2.3 Utilizzo di Directory e Elenchi di File
Lezione:	2 di 2

Introduzione

Il sistema operativo Unix è stato originariamente progettato per i computer mainframe a metà degli anni '60. Questi computer erano condivisi tra molti utenti che accedevano alle risorse di sistema tramite *terminali*. Queste idee di base sono ancora presenti nei sistemi Linux odierni. Parliamo ancora dell'utilizzo di "terminali" per inserire comandi nella shell; ogni sistema Linux è organizzato in modo tale che sia facile creare più utenti su un unico sistema.

Directory Home

Questo è un esempio di un normale file system in Linux:

```
$ tree -L 1 /
/
├── bin
├── boot
├── cdrom
├── dev
```

```
├── etc
├── home
├── lib
├── mnt
├── opt
├── proc
├── root
├── run
├── sbin
├── srv
├── sys
├── tmp
├── usr
└── var
```

La maggior parte di queste directory è presente su tutti i sistemi Linux. Dai server ai supercomputer, fino ai minuscoli sistemi integrati, un utente Linux esperto può essere sicuro di poter trovare il comando `ls` all'interno di `/bin`, di poter cambiare la configurazione del sistema modificando i file in `/etc` e di leggere i log di sistema in `/var`. La posizione standard di questi file e di queste directory è definita dal Filesystem Hierarchy Standard (FHS), che sarà discusso in una delle prossime lezioni. Imparerai di più sui contenuti di queste directory man mano che continuerai a studiare Linux, ma per il momento sappi che:

- le modifiche che fai nel filesystem root interessano tutti gli utenti
- la modifica dei file nel filesystem root richiede i permessi di amministratore.

Ciò significa che agli utenti normali è vietato modificare e forse persino leggere questi file. Tratteremo i permessi in una lezione successiva.

Ora concentriamoci sulla directory `/home`, che, a questo punto, dovrebbe essere piuttosto familiare:

```
$ tree -L 1 /home
/home
├── user
├── michael
└── lara
```

Il nostro sistema di esempio ha tre utenti normali e ciascuno di essi ha una propria area dedicata, dove può creare e modificare file e directory senza interferire con gli altri. Per esempio, nella lezione precedente stavamo lavorando con la seguente struttura di file:

```
$ tree /home/user
user
└── Documents
    ├── Mission-Statement
    └── Reports
        └── report2018.txt
```

In realtà, il vero filesystem potrebbe essere simile a questo:

```
$ tree /home
/home
├── user
│   └── Documents
│       ├── Mission-Statement
│       └── Reports
│           └── report2018.txt
├── michael
│   ├── Documents
│   │   └── presentation-for-clients.odp
│   └── Music
```

...e così via per `lara`.

In Linux `/home` è simile ad un condominio. Molti utenti possono avere qui il proprio spazio, che viene suddiviso in "appartamenti" dedicati. I servizi e la manutenzione dell'edificio stesso sono responsabilità dell'utente supervisore root.

Il Percorso Relativo Speciale per la Directory Home

Quando avvii una nuova sessione di terminale in Linux viene visualizzato un prompt dei comandi simile a questo:

```
user@hostname ~ $
```

La tilde (`~`) qui rappresenta la nostra *directory home*. Se esegui il comando `ls`, vedrai un output familiare:

```
$ cd ~
$ ls
```

```
Documents
```

Confrontalo con il file system sopra riportato per una migliore comprensione.

Riflettiamo ora su ciò che sappiamo di Linux: è simile a un condominio, con molti utenti che risiedono in `/home`. Quindi la home di `user` sarà diversa dalla home dell'utente `michael`. Per dimostrarlo, useremo il comando `su` che sta per *switch user*.

```
user@hostname ~ $ pwd
/home/user
user@hostname ~ $ su - michael
Password:
michael@hostname ~ $ pwd
/home/michael
```

Il significato di `~` cambia a seconda dell'utente. Per `michael`, il percorso assoluto di `~` è `/home/michael`. Per `lara`, il percorso assoluto di `~` è `/home/lara`, e così via.

Percorsi Relativi alla Home

Utilizzare `~` nei comandi è molto comodo, a patto che non si cambi utente. Esaminiamo il seguente esempio per `user`, che ha avviato una nuova sessione:

```
$ ls
Documents
$ cd Documents
$ ls
Mission-Statement
Reports
$ cd Reports
$ ls
report2018.txt
$ cd ~
$ ls
Documents
```

Nota che gli utenti iniziano sempre una nuova sessione nella loro directory home. In questo esempio, `user` ha navigato nella sua sottodirectory `Documents/Reports` e con il comando `cd ~` è tornato al punto di partenza. Puoi fare la stessa cosa usando il comando `cd` senza argomenti:

```
$ cd Documents/Reports
$ pwd
/home/user/Documents/Reports
$ cd
$ pwd
/home/user
```

Un'ultima cosa da notare: possiamo specificare le directory home di *altri utenti* indicando il nome utente dopo la tilde. Per esempio:

```
$ ls ~michael
Documents
Music
```

Nota che questo funzionerà solo se `michael` ci ha dato il permesso di visualizzare il contenuto della sua directory home.

Prendiamo ora in considerazione il caso in cui `michael` vorrebbe visualizzare il file `report2018.txt` nella directory home di `user`. Supponendo che `michael` sia autorizzato a farlo, può utilizzare il comando `less`.

```
$ less ~user/Documents/Reports/report2018.txt
```

Un qualsiasi percorso di file che contiene il carattere `~` è chiamato percorso *relativo alla home*.

File e Directory Nascosti

Nella lezione precedente abbiamo introdotto l'opzione `-a` per il comando `ls`. Abbiamo usato `ls -a` per presentare i due percorsi relativi speciali: `.` e `..`. L'opzione `-a` elenca *tutti* i file e le directory, inclusi i file e le directory *nascoste*.

```
$ ls -a ~
.
..
.bash_history
.bash_logout
.bash-profile
.bashrc
Documents
```

I file e le directory nascoste iniziano sempre con un punto (`.`). Per impostazione predefinita, la directory home di un utente include molti file nascosti. Questi sono spesso usati per stabilire impostazioni di configurazione specifiche dell'utente e dovrebbero essere modificati solo da un utente esperto.

L'Opzione Formato di Elenco Lungo (Long List)

Il comando `ls` ha molte opzioni a disposizione. Diamo un'occhiata a una delle opzioni più comuni:

```
$ ls -l
-rw-r--r-- 1 user staff      3606 Jan 13  2017 report2018.txt
```

`-l` crea un *elenco lungo* (long list). I file e le directory occuperanno una riga ciascuno, ma verranno visualizzate anche informazioni aggiuntive per ciascuno di essi.

`-rw-r--r--`

Tipo di file e permessi del file. Nota che un file normale inizia con un trattino mentre una directory inizia con `d`.

`1`

Numero di link al file.

`user staff`

Specifica chi possiede il file (ownership). `user` è il proprietario del file e il file è anche associato al gruppo `staff`.

`3606`

Dimensioni del file in byte.

`Jan 13 2017`

Time stamp dell'ultima modifica al file.

`report2018.txt`

Nome del file.

Argomenti come ownership, permessi e link verranno trattati nelle lezioni successive. Come puoi vedere, la variante di `ls` con il formato di elenco lungo (long list) è spesso preferibile a quella predefinita.

Altre Opzioni di ls

Di seguito sono riportati alcuni degli utilizzi più comuni del comando `ls`. Come puoi vedere l'utente può combinare molte opzioni insieme per ottenere l'output desiderato.

`ls -lh`

La combinazione dell'opzione *formato di elenco lungo* (long list) con l'opzione dimensioni di file *leggibili dall'uomo* (human readable) ci fornisce utili suffissi, come per esempio `M` per megabyte o `K` per kilobyte.

`ls -d */`

L'opzione `-d` elenca le directory ma non il loro contenuto. Combinato con `*/` mostra solo le sottodirectory e nessun file.

`ls -lt`

Combina il *formato di elenco lungo* (long list) con l'opzione di ordinamento per *data di modifica* (modification time). I file con le modifiche più recenti saranno visualizzati in alto, mentre i file con le modifiche meno recenti saranno visualizzati in fondo. Ma questo ordine può essere invertito con:

`ls -lrt`

Combina il *formato di elenco lungo* (long list) con *l'ordinamento per data di modifica* (modification time) e con `-r` che *inverte* (reverse) l'ordinamento. Ora i file con le modifiche più recenti si troveranno in fondo all'elenco. Oltre all'ordinamento per *data di modifica* (modification time), i file possono essere ordinati anche per *data di accesso* (access time) o per *data di modifica dello stato* (status time).

`ls -lX`

Combina il *formato di elenco lungo* (long list) con l'opzione di ordinamento per *estesione dei file* (file eXtension). Questo criterio, per esempio, raggrupperà tutti i file che terminano con `.txt`, tutti i file che terminano con `.jpg` e così via.

`ls -S`

L'opzione `-S` ordina per *dimensione* (size) dei file, proprio come `-t` e `-X` ordinano rispettivamente per data di modifica ed estensione. I file più grandi verranno visualizzati per primi mentre quelli più piccoli verranno visualizzati per ultimi. Nota che il contenuto delle sottodirectory *non* è incluso nell'ordinamento.

`ls -R`

L'opzione `-R` modifica il comando `ls` per visualizzare un elenco *elenco ricorsivo*. Che cosa

significa?

La Ricorsione in Bash

La ricorsione si riferisce a una situazione in cui "qualcosa è definito in relazione a se stesso". La ricorsione è un concetto molto importante nell'informatica, ma qui il suo significato è molto più semplice. Consideriamo il nostro esempio di prima:

```
$ ls ~
Documents
```

Sappiamo già che `user` ha una directory home e che in questa directory c'è una sottodirectory. Finora `ls` ci ha mostrato solamente i file e le sottodirectory di un certo percorso, senza però indicarci il contenuto di queste sottodirectory. In queste lezioni abbiamo usato il comando `tree` per visualizzare il contenuto di molte directory. Sfortunatamente `tree` non è uno degli strumenti di base di Linux e quindi non è sempre disponibile. Confronta l'output di `tree` con l'output di `ls -R` nei seguenti esempi:

```
$ tree /home/user
user
└── Documents
    ├── Mission-Statement
    └── Reports
        └── report2018.txt

$ ls -R ~
/home/user/:
Documents

/home/user/Documents:
Mission-Statement
Reports

/home/user/Documents/Reports:
report2018.txt
```

Con l'opzione ricorsiva otteniamo quindi un elenco di file molto più lungo. In effetti è come se avessimo eseguito il comando `ls` nella directory home di `user` e avessimo incontrato una sottodirectory. Quindi siamo entrati in questa sottodirectory e abbiamo eseguito di nuovo il comando `ls`. Abbiamo incontrato il file `Mission-Statement` e un'altra sottodirectory chiamata

`Reports`. E ancora, siamo entrati nella sottodirectory e abbiamo eseguito di nuovo il comando `ls`. Essenzialmente, eseguire `ls -R` è come dire a Bash: “Esegui `ls` qui e ripeti il comando in ogni sottodirectory che trovi.”

La ricorsione è particolarmente importante nei comandi di modifica dei file, come per esempio per copiare o rimuovere directory. Per esempio, se si desidera copiare la sottodirectory `Documents`, sarà necessario specificare una copia ricorsiva per estendere questo comando a tutte le sottodirectory.

Esercizi Guidati

1. Utilizza la seguente struttura di file per rispondere alle prossime tre domande:

```
/
├── etc/
│   ├── network/
│   │   └── interfaces/
│   ├── systemd/
│   │   ├── resolved.conf
│   │   ├── system/
│   │   ├── system.conf
│   │   ├── user/
│   │   └── user.conf
│   └── udev/
│       ├── rules.d
│       └── udev.conf
└── home/
    ├── lost+found/
    ├── user/
    │   └── Documents/
    └── michael/
        └── Music/
```

   - Quale comando ti consente di navigare nella directory `network` indipendentemente dalla tua posizione corrente?

   - Quale comando può inserire `user` per navigare nella propria directory `Documents` da `/etc/udev`? Usa il percorso più breve possibile.

   - Quale comando può inserire `user` per navigare nella directory `Music` di `michael`? Usa il percorso più breve possibile.

2. Considera il seguente output di `ls -lh` per rispondere alle prossime due domande. Nota che le directory sono indicate con una `d` all'inizio della riga.

```
drwxrwxrwx  5 eric eric  4.0K Apr 26  2011 China/
-rwxrwxrwx  1 eric eric  1.5M Jul 18  2011 img_0066.jpg
```

```
-rwxrwxrwx  1 eric eric  1.5M Jul 18  2011 img_0067.jpg
-rwxrwxrwx  1 eric eric  1.6M Jul 18  2011 img_0074.jpg
-rwxrwxrwx  1 eric eric  1.8M Jul 18  2011 img_0075.jpg
-rwxrwxrwx  1 eric eric   46K Jul 18  2011 scary.jpg
-rwxrwxrwx  1 eric eric  469K Jan 29  2018 Screenshot from 2017-08-13 21-22-24.png
-rwxrwxrwx  1 eric eric  498K Jan 29  2018 Screenshot from 2017-08-14 21-18-07.png
-rwxrwxrwx  1 eric eric  211K Jan 29  2018 Screenshot from 2018-01-06 23-29-30.png
-rwxrwxrwx  1 eric eric  150K Jul 18  2011 tobermory.jpg
drwxrwxrwx  6 eric eric  4.0K Apr 26  2011 Tokyo/
-rwxrwxrwx  1 eric eric  1.4M Jul 18  2011 Toronto 081.jpg
-rwxrwxrwx  1 eric eric  1.4M Jul 18  2011 Toronto 085.jpg
-rwxrwxrwx  1 eric eric  944K Jul 18  2011 Toronto 152.jpg
-rwxrwxrwx  1 eric eric  728K Jul 18  2011 Toronto 173.jpg
drwxrwxrwx  2 eric eric  4.0K Jun  5  2016 Wallpapers/
```

- Quale file apparirà all'inizio se si esegue il comando `ls -lrS`?

- Descrivi ciò che ti aspetti di vedere come output di `ls -ad */`.

Esercizi Esplorativi

1. Esegui il comando `ls -lh` in una directory che contiene sottodirectory. Nota la dimensione riportata per queste directory. Ti sembrano corrette queste dimensioni? Corrispondono esattamente al contenuto di tutti i file all'interno della directory?

2. Ecco un nuovo comando da provare: `du -h`. Esegui questo comando e descrivine l'output.

3. Su molti sistemi Linux puoi digitare `ll` e ottenere lo stesso output che avresti digitando `ls -l`. Nota tuttavia che `ll` *non* è un comando. Per esempio, `man ll` ti mostrerà il messaggio che non esiste alcuna pagina di manuale per `ll`. Questo è un esempio di *alias*. Perché gli alias potrebbero essere utili a un utente?

Sommario

In questa lezione hai imparato:

- che ogni utente Linux ha una directory home;
- che è possibile accedere alla directory home dell'utente corrente utilizzando `~`;
- che qualsiasi percorso di file che utilizza `~` è chiamato percorso *relativo alla home*.

Hai anche imparato alcuni dei modi più comuni per modificare il comando `ls`.

`-a` (all)
stampa tutti i file/directory, inclusi quelli nascosti

`-d` (directories)
elenca le directory, non il loro contenuto

`-h` (human readable)
stampa le dimensioni dei file in un formato leggibile dall'uomo

`-l` (long list)
fornisce dettagli extra, mostrando un file/directory per riga

`-r` (reverse)
inverte l'ordine di una selezione

`-R` (recursive)
elenca tutti i file, inclusi i file in ogni sottodirectory

`-S` (size)
ordina per dimensione dei file

`-t` (time)
ordina per data di modifica

`-X` (eXtension)
ordina per estensione dei file

Risposte agli Esercizi Guidati

1. Utilizza la seguente struttura di file per rispondere alle prossime tre domande:

```
/
├── etc/
│   ├── network/
│   │   └── interfaces/
│   ├── systemd/
│   │   ├── resolved.conf
│   │   ├── system/
│   │   ├── system.conf
│   │   ├── user/
│   │   └── user.conf
│   └── udev/
│       ├── rules.d
│       └── udev.conf
└── home/
    ├── lost+found/
    ├── user/
    │   └── Documents/
    └── michael/
        └── Music/
```

   - Quale comando ti consente di navigare nella directory `network` indipendentemente dalla tua posizione corrente?

     ```
     cd /etc/network
     ```

   - Quale comando può inserire `user` per navigare nella propria directory `Documents` da `/etc/udev`? Usa il percorso più breve possibile.

     ```
     cd ~/Documents
     ```

   - Quale comando può inserire `user` per navigare nella directory `Music` di `michael`? Usa il percorso più breve possibile.

     ```
     cd ~michael/Music
     ```

2. Considera il seguente output di `ls -lh` per rispondere alle prossime due domande. Nota che le directory sono indicate con una `d` all'inizio della riga.

```
drwxrwxrwx  5 eric eric  4.0K Apr 26  2011 China/
-rwxrwxrwx  1 eric eric  1.5M Jul 18  2011 img_0066.jpg
-rwxrwxrwx  1 eric eric  1.5M Jul 18  2011 img_0067.jpg
-rwxrwxrwx  1 eric eric  1.6M Jul 18  2011 img_0074.jpg
-rwxrwxrwx  1 eric eric  1.8M Jul 18  2011 img_0075.jpg
-rwxrwxrwx  1 eric eric   46K Jul 18  2011 scary.jpg
-rwxrwxrwx  1 eric eric  469K Jan 29  2018 Screenshot from 2017-08-13 21-22-24.png
-rwxrwxrwx  1 eric eric  498K Jan 29  2018 Screenshot from 2017-08-14 21-18-07.png
-rwxrwxrwx  1 eric eric  211K Jan 29  2018 Screenshot from 2018-01-06 23-29-30.png
-rwxrwxrwx  1 eric eric  150K Jul 18  2011 tobermory.jpg
drwxrwxrwx  6 eric eric  4.0K Apr 26  2011 Tokyo/
-rwxrwxrwx  1 eric eric  1.4M Jul 18  2011 Toronto 081.jpg
-rwxrwxrwx  1 eric eric  1.4M Jul 18  2011 Toronto 085.jpg
-rwxrwxrwx  1 eric eric  944K Jul 18  2011 Toronto 152.jpg
-rwxrwxrwx  1 eric eric  728K Jul 18  2011 Toronto 173.jpg
drwxrwxrwx  2 eric eric  4.0K Jun  5  2016 Wallpapers/
```

   - Quale file apparirà all'inizio se si esegue il comando `ls -lrS`?

     Le tre cartelle sono tutte di 4.0K, che è la dimensione più piccola. `ls` ordinerà poi le directory in ordine alfabetico per impostazione predefinita. La risposta corretta è il file `scary.jpg`.

   - Descrivi ciò che ti aspetti di vedere come output di `ls -ad */`.

     Questo comando mostrerà tutte le sottodirectory, comprese le sottodirectory nascoste.

Risposte agli Esercizi Esplorativi

1. Esegui il comando `ls -lh` in una directory che contiene sottodirectory. Nota la dimensione riportata per queste directory. Ti sembrano corrette queste dimensioni? Corrispondono esattamente al contenuto di tutti i file all'interno della directory?

   No. Ogni directory ha una dimensione di 4096 byte. Questo perché le directory qui sono un'astrazione: non esistono come una struttura ad albero sul disco. Quando trovi elencata una directory, vedi un *collegamento* a un elenco di file. La dimensione di questi collegamenti è di 4096 byte.

2. Ecco un nuovo comando da provare: `du -h`. Esegui questo comando e descrivine l'output.

   Il comando `du` genera un elenco di tutti i file e di tutte le directory, indicando la dimensione di ciascuno di essi. Per esempio, `du -s` mostra la dimensione di tutti i file, di tutte le directory e di tutte le sottodirectory in una determinata posizione.

3. Su molti sistemi Linux puoi digitare `ll` e ottenere lo stesso output che avresti digitando `ls -l`. Nota tuttavia che `ll` *non* è un comando. Per esempio, `man ll` ti mostrerà il messaggio che non esiste alcuna pagina di manuale per `ll`. Questo è un esempio di *alias*. Perché gli alias potrebbero essere utili a un utente?

   `ll` è un *alias* di `ls -l`. In Bash, possiamo usare gli alias per semplificare i comandi usati più frequentemente. `ll` è spesso già definito in Linux, ma puoi anche creare anche un tuo personale alias.

2.4 Creazione, Spostamento ed Eliminazione di File

Obiettivi LPI di riferimento

Linux Essentials version 1.6, Exam 010, Objective 2.4

Peso

2

Aree di Conoscenza Chiave

- File e directory
- Sensibilità a minuscole e maiuscole
- Semplici raggruppamenti (globbing)

Di seguito è riportato un elenco parziale dei file, dei termini e dei comandi utilizzati

- `mv`, `cp`, `rm`, `touch`
- `mkdir`, `rmdir`

2.4 Lezione 1

Certificazione:	Linux Essentials
Versione:	1.6
Argomento:	2 Trovare il Proprio Modo di Operare su un Sistema Linux
Obiettivo:	2.4 Creazione, Spostamento ed Eliminazione di File
Lezione:	1 di 1

Introduzione

Questa lezione tratta la gestione di file e directory su Linux attraverso strumenti a Command Line.

Un file è una raccolta di dati con un nome e un insieme di attributi. Se, per esempio, dovessi trasferire alcune foto dal tuo telefono a un computer e dovessi dar loro dei nomi descrittivi, ora avresti, su questo computer, un mucchio di file di immagine. Questi file hanno degli attributi come, per esempio, la data dell'ultimo accesso o dell'ultima modifica al file.

Una directory è un tipo speciale di file utilizzato per organizzare i file. Un buon modo per pensare alle directory consiste nel paragonarle alle cartelline utilizzate per organizzare i documenti in uno schedario. A differenza delle cartelline cartacee, puoi facilmente inserire delle directory all'interno di altre directory.

La Command Line è il mezzo più efficace per gestire i file su un sistema Linux. Gli strumenti della shell e della Command Line hanno caratteristiche che rendono l'uso della Command Line più veloce e più semplice di un file manager grafico.

In questa lezione userai i comandi `ls`, `mv`, `cp`, `pwd`, `find`, `touch`, `rm`, `rmdir`, `echo`, `cat` e `mkdir` per gestire e organizzare file e directory.

Distinzione tra Maiuscole e Minuscole

A differenza di Microsoft Windows, i nomi di file e directory sui sistemi Linux fanno distinzione tra maiuscole e minuscole. Ciò significa che i nomi `/etc/` ed `/ETC/` rappresentano differenti directory. Prova i seguenti comandi:

```
$ cd /
$ ls
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
$ cd ETC
bash: cd: ETC: No such file or directory
$ pwd
/
$ cd etc
$ pwd
/etc
```

Il comando `pwd` mostra la directory in cui ti trovi attualmente. Come puoi vedere il tentativo di spostarti in `/ETC` non ha funzionato in quanto non esiste tale directory. Lo spostamento nella directory `/etc`, che esiste, è invece andato a buon file.

Creare Directory

Il comando `mkdir` viene utilizzato per creare directory.

Creiamo una nuova directory all'interno della nostra directory home:

```
$ cd ~
$ pwd
/home/emma
$ ls
Desktop  Documents  Downloads
$ mkdir linux_essentials-2.4
$ ls
Desktop  Documents  Downloads  linux_essentials-2.4
$ cd linux_essentials-2.4
$ pwd
```

```
/home/emma/linux_essentials-2.4
```

Nel corso di questa lezione eseguiremo tutti i comandi all'interno di questa directory o in una delle sue sottodirectory.

Per tornare facilmente alla directory della lezione da qualsiasi altra posizione nel tuo file system, puoi utilizzare il comando:

```
$ cd ~/linux_essentials-2.4
```

La shell interpreta il carattere `~` come la tua directory home.

Quando sei nella directory della lezione, crea altre directory da utilizzare per gli esercizi. Puoi aggiungere tutti i nomi delle directory, separati da spazi, al comando `mkdir`:

```
$ mkdir creating moving copying/files copying/directories deleting/directories
deleting/files globs
mkdir: cannot create directory ‘copying/files’: No such file or directory
mkdir: cannot create directory ‘copying/directories’: No such file or directory
mkdir: cannot create directory ‘deleting/directories’: No such file or directory
mkdir: cannot create directory ‘deleting/files’: No such file or directory
$ ls
creating  globs  moving
```

Nota il messaggio di errore e nota che sono state create solo `moving`, `globs` e `creating`. Le directory `copying` e `deleting` non esistono ancora. `mkdir`, per impostazione predefinita, non crea una directory all'interno di una directory che non esiste. L'opzione `-p` o `--parents` indica a `mkdir` di creare le directory padre se non esistono. Prova lo stesso comando `mkdir` con l'opzione `-p`:

```
$ mkdir -p creating moving copying/files copying/directories deleting/directories
deleting/files globs
```

Questa volta non hai ricevuto alcun messaggio di errore. Vediamo quali directory esistono ora:

```
$ find
.
./creating
./moving
./globs
```

```
./copying
./copying/files
./copying/directories
./deleting
./deleting/directories
./deleting/files
```

Il programma `find` è solitamente usato per cercare file e directory, ma senza alcuna opzione ti mostra un elenco di tutti i file, di tutte le directory e di tutte le sottodirectory della tua directory corrente.

TIP Quando si elenca il contenuto di una directory con `ls`, le opzioni `-t` e `-r` sono particolarmente utili. Ordinano l'output per data (`-t`) ed invertono l'ordinamento (`-r`). In questo caso, i file più recenti saranno in fondo all'output.

Creare i File

In genere i file vengono creati dai programmi, i quali lavorano con i dati memorizzati nei file. Un file vuoto può essere creato usando il comando `touch`. Se esegui `touch` su un file esistente, il contenuto del file non verrà modificato, ma il timestamp di modifica del file verrà aggiornato.

Esegui il seguente comando per creare alcuni file per la lezione di globbing:

```
$ touch globs/question1 globs/question2012 globs/question23 globs/question13
globs/question14
$ touch globs/star10 globs/star1100 globs/star2002 globs/star2013
```

Ora verifichiamo che tutti i file esistano nella directory `globs`:

```
$ cd globs
$ ls
question1   question14    question23  star1100  star2013
question13  question2012  star10      star2002
```

Hai visto come `touch` ha creato i file? Puoi visualizzare il contenuto di un file di testo con il comando `cat`. Provalo su uno dei file che hai appena creato:

```
$ cat question14
```

Poiché `touch` crea file vuoti, non dovresti ottenere alcun output. Puoi usare `echo` con `>` per creare

semplici file di testo. Provalo:

```
$ echo hello > question15
$ cat question15
hello
```

`echo` visualizza il testo sulla Command Line. Il carattere `>` indica alla shell di scrivere l'output di un comando nel file specificato invece che sul terminale. Questo fa sì che l'output di `echo`, `hello` in questo caso, venga scritto nel file `question15`. Non è specifico di `echo`, ma può essere utilizzato con qualsiasi comando.

WARNING Fai attenzione quando usi `>`! Se il file indicato esiste già, verrà sovrascritto!

Rinominare i File

I file vengono spostati e rinominati con il comando `mv`.

Imposta `moving` come directory di lavoro:

```
$ cd ~/linux_essentials-2.4/moving
```

Crea alcuni file con cui esercitarti. A questo punto dovresti già avere familiarità con questi comandi:

```
$ touch file1 file22
$ echo file3 > file3
$ echo file4 > file4
$ ls
file1  file22  file3  file4
```

Supponiamo che `file22` sia un errore di battitura e che il nome corretto dovesse essere `file2`. Correggilo con il comando `mv`. Quando si rinomina un file, il primo argomento è il nome corrente, mentre il secondo è il nuovo nome:

```
$ mv file22 file2
$ ls
file1  file2  file3  file4
```

Fai attenzione con il comando `mv`. Se rinomini un file con il nome di un file esistente, quest'ultimo

verrà sovrascritto. Facciamo una prova con `file3` e `file4`:

```
$ cat file3
file3
$ cat file4
file4
$ mv file4 file3
$ cat file3
file4
$ ls
file1  file2  file3
```

Nota come il contenuto di `file3` sia ora `file4`. Usa l'opzione `-i` per fare in modo che `mv` ti avverta se stai per sovrascrivere un file esistente. Provalo:

```
$ touch file4 file5
$ mv -i file4 file3
mv: overwrite ‘file3’? y
```

Spostare i File

I file vengono spostati da una directory all'altra con il comando `mv`.

Crea alcune directory in cui spostare i file:

```
$ cd ~/linux_essentials-2.4/moving
$ mkdir dir1 dir2
$ ls
dir1  dir2  file1  file2  file3  file5
```

Sposta `file1` in `dir1`:

```
$ mv file1 dir1
$ ls
dir1  dir2  file2  file3  file5
$ ls dir1
file1
```

Nota che l'ultimo argomento di `mv` è la directory di destinazione. Se l'ultimo argomento di `mv` è

una directory, i file vengono spostati al suo interno. È possibile specificare più file in un singolo comando `mv`:

```
$ mv file2 file3 dir2
$ ls
dir1  dir2  file5
$ ls dir2
file2  file3
```

È anche possibile usare `mv` per spostare e rinominare directory. Rinomina `dir1` in `dir3`:

```
$ ls
dir1  dir2  file5
$ ls dir1
file1
$ mv dir1 dir3
$ ls
dir2  dir3  file5
$ ls dir3
file1
```

Eliminare File e Directory

Il comando `rm` può eliminare file e directory, mentre il comando `rmdir` può eliminare solo directory. Puliamo la directory `moving` eliminando `file5`:

```
$ cd ~/linux_essentials-2.4/moving
$ ls
dir2  dir3  file5
$ rmdir file5
rmdir: failed to remove ‘file5’: Not a directory
$ rm file5
$ ls
dir2  dir3
```

Per impostazione predefinita, `rmdir` può eliminare solo directory vuote; quindi abbiamo dovuto usare `rm` per eliminare un file normale. Prova a eliminare la directory `deleting`:

```
$ cd ~/linux_essentials-2.4/
$ ls
```

```
copying  creating  deleting  globs  moving
$ rmdir deleting
rmdir: failed to remove ‘deleting’: Directory not empty
$ ls -l deleting
total 0
drwxrwxr-x. 2 emma emma 6 Mar 26 14:58 directories
drwxrwxr-x. 2 emma emma 6 Mar 26 14:58 files
```

Per impostazione predefinita, `rmdir` rifiuta di eliminare una directory che non è vuota. Usa `rmdir` per rimuovere una delle sottodirectory vuote della directory `deleting`:

```
$ ls -a deleting/files
.  ..
$ rmdir deleting/files
$ ls -l deleting
directories
```

Eliminare un gran numero di file o strutture profonde di directory con molte sottodirectory può sembrare noioso, ma in realtà è semplice. Per impostazione predefinita, `rm` funziona solo con file normali. L'opzione `-r` è utilizzata per sovrascrivere questo comportamento. Fai attenzione, `rm -r` è estremamente pericoloso! Quando usi l'opzione `-r`, `rm` non solo cancella qualsiasi directory, ma anche tutto ciò che si trova all'interno di quella directory, incluse le sottodirectory e il loro contenuto. Guarda tu stesso come funziona `rm -r`:

```
$ ls
copying  creating  deleting  globs  moving
$ rm deleting
rm: cannot remove ‘deleting’: Is a directory
$ ls -l deleting
total 0
drwxrwxr-x. 2 emma emma 6 Mar 26 14:58 directories
$ rm -r deleting
$ ls
copying  creating  globs  moving
```

Hai notato come `deleting` sia scomparsa, anche se non era vuota? Come `mv`, `rm` ha un'opzione `-i` per chiedere conferma prima di fare qualsiasi cosa. Usa `rm -ri` per rimuovere da `moving` le directory che non sono più necessarie:

```
$ find
```

```
.
./creating
./moving
./moving/dir2
./moving/dir2/file2
./moving/dir2/file3
./moving/dir3
./moving/dir3/file1
./globs
./globs/question1
./globs/question2012
./globs/question23
./globs/question13
./globs/question14
./globs/star10
./globs/star1100
./globs/star2002
./globs/star2013
./globs/question15
./copying
./copying/files
./copying/directories
$ rm -ri moving
rm: descend into directory ‘moving’? y
rm: descend into directory ‘moving/dir2’? y
rm: remove regular empty file ‘moving/dir2/file2’? y
rm: remove regular empty file ‘moving/dir2/file3’? y
rm: remove directory ‘moving/dir2’? y
rm: descend into directory ‘moving/dir3’? y
rm: remove regular empty file ‘moving/dir3/file1’? y
rm: remove directory ‘moving/dir3’? y
rm: remove directory ‘moving’? y
```

Copiare File e Directory

Il comando `cp` viene utilizzato per copiare file e directory. Copia alcuni file nella directory `copying`:

```
$ cd ~/linux_essentials-2.4/copying
$ ls
directories  files
$ cp /etc/nsswitch.conf files/nsswitch.conf
```

```
$ cp /etc/issue /etc/hostname files
```

Se l'ultimo argomento è una directory, `cp` crea una copia degli argomenti precedenti all'interno della directory. Come `mv`, è possibile specificare più file contemporaneamente, purché la destinazione sia una directory.

Quando entrambi gli operandi di `cp` sono file esistenti, `cp` sovrascrive il secondo file con una copia del primo file. Facciamo pratica sovrascrivendo il file `issue` con il file `hostname`:

```
$ cd ~/linux_essentials-2.4/copying/files
$ ls
hostname  issue  nsswitch.conf
$ cat hostname
mycomputer
$ cat issue
Debian GNU/Linux 9 \n \l

$ cp hostname issue
$ cat issue
mycomputer
```

Ora proviamo a creare una copia della directory `files` all'interno della directory `directories`:

```
$ cd ~/linux_essentials-2.4/copying
$ cp files directories
cp: omitting directory 'files'
```

Come puoi vedere, `cp`, per impostazione predefinita, funziona solo su singoli file. Per copiare una directory, usa l'opzione `-r`. Tieni presente che l'opzione `-r` farà sì che `cp` copi anche il contenuto della directory che stai copiando:

```
$ cp -r files directories
$ find
.
./files
./files/nsswitch.conf
./files/fstab
./files/hostname
./directories
./directories/files
./directories/files/nsswitch.conf
```

```
./directories/files/fstab
./directories/files/hostname
```

Hai notato che, quando una directory esistente viene usata come destinazione, `cp` crea una copia della directory sorgente al suo interno? Se la destinazione non esiste, viene creata e riempita con il contenuto della directory sorgente:

```
$ cp -r files files2
$ find
.
./files
./files/nsswitch.conf
./files/fstab
./files/hostname
./directories
./directories/files
./directories/files/nsswitch.conf
./directories/files/fstab
./directories/files/hostname
./files2
./files2/nsswitch.conf
./files2/fstab
./files2/hostname
```

Globbing

Ciò che viene comunemente chiamato globbing è un semplice linguaggio di corrispondenza di pattern. Le shell a Command Line sui sistemi Linux utilizzano questo linguaggio per fare riferimento a gruppi di file i cui nomi hanno corrispondenza con uno specifico pattern. Lo standard POSIX.1-2017 definisce i seguenti caratteri per la corrispondenza con i pattern:

`*`
: Corrisponde a un qualsiasi numero di un qualsiasi carattere, incluso nessun carattere

`?`
: Corrisponde a un qualsiasi carattere

`[]`
: Corrisponde a una classe di caratteri

Tradotto, questo significa che puoi dire alla tua shell di cercare corrispondenze in base a un

pattern invece che in base a una stringa di testo letterale. Di solito gli utenti Linux specificano più file con un glob invece di digitare singolarmente ogni nome di file. Esegui i seguenti comandi:

```
$ cd ~/linux_essentials-2.4/globs
$ ls
question1   question14  question2012  star10    star2002
question13  question15  question23    star1100  star2013
$ ls star1*
star10  star1100
$ ls star*
star10  star1100  star2002  star2013
$ ls star2*
star2002  star2013
$ ls star2*2
star2002
$ ls star2013*
star2013
```

La shell espande `*` a *qualsiasi numero di qualsiasi cosa*; quindi la tua shell interpreta `star*` come qualsiasi cosa, pertinente al contesto, che inizia con `star`. Quando esegui il comando `ls star*`, la tua shell non esegue il programma `ls` con un argomento `star*`, ma cerca i file nella directory corrente che corrispondono al pattern `star*` (incluso solo `star`) e trasforma ogni file che cha corrispondenza con il pattern in un argomento di `ls`:

```
$ ls star*
```

Per quanto riguarda `ls`, è l'equivalente di

```
$ ls star10  star1100  star2002  star2013
```

Il carattere `*` non ha alcun significato per `ls`. Per dimostrarlo, esegui il seguente comando:

```
$ ls star\*
ls: cannot access star*: No such file or directory
```

Quando precedi un carattere con `\`, stai dicendo alla tua shell di non interpretarlo. In questo caso, vuoi che `ls` abbia un argomento `star*` invece dell'espansione fatta dal glob `star*`.

Il `?` si espande in *ogni singolo carattere*. Prova i seguenti comandi per testarlo tu stesso:

```
$ ls
question1   question14  question2012  star10    star2002
question13  question15  question23    star1100  star2013
$ ls question?
question1
$ ls question1?
question13  question14  question15
$ ls question?3
question13  question23
$ ls question13?
ls: cannot access question13?: No such file or directory
```

Le parentesi quadre `[]` vengono utilizzate per indicare la corrispondenza con intervalli o classi di caratteri. Le parentesi quadre `[]` funzionano come nelle espressioni regolari POSIX con l'eccezione che i glob usano `^` al posto di `!`.

Crea alcuni file con cui sperimentare:

```
$ mkdir brackets
$ cd brackets
$ touch file1 file2 file3 file4 filea fileb filec file5 file6 file7
```

Gli intervalli all'interno delle parentesi quadre `[]` sono espressi utilizzando un `-`:

```
$ ls
file1  file2  file3  file4  file5  file6  file7  filea  fileb  filec
$ ls file[1-2]
file1  file2
$ ls file[1-3]
file1  file2  file3
```

È possibile specificare più intervalli:

```
$ ls file[1-25-7]
file1  file2  file5  file6  file7
$ ls file[1-35-6a-c]
file1  file2  file3  file5  file6  filea  fileb  filec
```

Le parentesi quadre possono essere utilizzate anche per indicare la corrispondenza con un insieme specifico di caratteri.

```
$ ls file[1a5]
file1  file5  filea
```

Puoi anche usare il carattere `^` come primo carattere per indicare la corrispondenza con qualsiasi valore tranne alcuni caratteri.

```
$ ls file[^a]
file1  file2  file3  file4  file5  file6  file7  fileb  filec
```

Infine, in questa lezione, tratteremo le classi di caratteri. Per indicare la corrispondenza con una classe di caratteri, utilizza `[:classname:]`. Per esempio, per utilizzare la classe digit, che corrisponde ai numeri, dovresti fare qualcosa del genere:

```
$ ls file[[:digit:]]
file1  file2  file3  file4  file5  file6  file7
$ touch file1a file11
$ ls file[[:digit:]a]
file1  file2  file3  file4  file5  file6  file7  filea
$ ls file[[:digit:]]a
file1a
```

Il glob `file[[:digit:]a]`, corrisponde a `file` seguito da un numero o da una `a`.

Le classi di caratteri supportate dipendono dal Locale impostato. POSIX richiede le seguenti classi di caratteri per tutti i Locale:

`[:alnum:]`
: Lettere e numeri.

`[:alpha:]`
: Lettere maiuscole o minuscole.

`[:blank:]`
: Spazi e tabulazioni.

`[:cntrl:]`
: Caratteri di controllo, per esempio backspace, campanello, NAK, escape.

`[:digit:]`
Numeri (`0123456789`).

`[:graph:]`
Caratteri grafici (tutti i caratteri tranne `ctrl` e il carattere spazio).

`[:lower:]`
Lettere minuscole (`a-z`).

`[:print:]`
Caratteri stampabili (`alnum`, `punct` e il carattere spazio).

`[:punct:]`
Caratteri di punteggiatura, per esempio `!`, `&`, `"`.

`[:space:]`
Caratteri spazi bianchi, per esempio tabulazioni, spazi, nuove righe.

`[:upper:]`
Lettere maiuscole (`A-Z`).

`[:xdigit:]`
Numeri esadecimali (solitamente `0123456789abcdefABCDEF`).

Esercizi Guidati

1. Tenendo presente quanto sotto riportato, seleziona le directory che verrebbero create con il comando `mkdir -p /tmp/outfiles/text/today /tmp/infiles/text/today`

```
$ pwd
/tmp
$ find
.
./outfiles
./outfiles/text
```

`/tmp`	
`/tmp/outfiles`	
`/tmp/outfiles/text`	
`/tmp/outfiles/text/today`	
`/tmp/infiles`	
`/tmp/infiles/text`	
`/tmp/infiles/text/today`	

2. Che cosa fa `-v` nei comandi `mkdir`, `rm` e `cp`?

3. Che cosa succede se si tenta accidentalmente di copiare tre file sulla stessa Command Line in un file già esistente anziché in una directory?

4. Che cosa succede quando usi `mv` per spostare una directory dentro se stessa?

5. Come potresti eliminare tutti i file nella directory corrente che iniziano con `old`?

6. Quale dei seguenti file corrisponde a `log_[a-z]_201?_*_01.txt`?

`log_3_2017_Jan_01.txt`	
`log_+_2017_Feb_01.txt`	

`log_b_2007_Mar_01.txt`	
`log_f_201A_Wednesday_01.txt`	

7. Indica alcuni glob che abbiano corrispondenza con tutti i file in elenco:

```
doc100
doc200
doc301
doc401
```

Esercizi Esplorativi

1. Usa la pagina man di `cp` per scoprire come fare la copia di un file mantenendo i permessi e la data di modifica del file originale.

2. Che cosa fa il comando `rmdir -p`? Provalo e spiega come si differenzia da `rm -r`.

3. NON ESEGUIRE REALMENTE QUESTO COMANDO: Cosa pensi che farebbe `rm -ri /*`? (SUL SERIO, NON PROVARE A ESEGUIRLO!)

4. Oltre a usare `-i`, è possibile impedire a `mv` di sovrascrivere i file di destinazione?

5. Spiega il comando `cp -u`.

 Questa opera è concessa in licenza CC BY-NC-ND 4.0. |

Sommario

L'ambiente Linux a Command Line fornisce strumenti per gestire i file. Alcuni di uso comune sono: `cp`, `mv`, `mkdir`, `rm` e `rmdir`. Questi strumenti, combinati con i glob, consentono agli utenti di svolgere tante attività in tempi molto brevi.

Molti comandi hanno un'opzione `-i`, che chiede conferma prima di fare qualsiasi cosa. Questa conferma può evitarti molti problemi se hai digitato qualcosa di sbagliato.

Molti comandi hanno un'opzione `-r`. L'opzione `-r` di solito significa ricorsione. In matematica e informatica una funzione ricorsiva è una funzione che utilizza se stessa nella sua definizione. Quando si tratta di strumenti a Command Line di solito significa applicare il comando a una directory e a tutto ciò che contiene.

Comandi utilizzati in questa lezione:

`cat`
: Legge e visualizza il contenuto di un file.

`cp`
: Copia file o directory.

`echo`
: Produce una stringa come output.

`find`
: Attraversa un albero del file system e cerca i file che corrispondono a un insieme specifico di criteri.

`ls`
: Mostra le proprietà di file e directory ed elenca il contenuto di una directory.

`mkdir`
: Crea nuove directory.

`mv`
: Sposta o rinomina file o directory.

`pwd`
: Visualizza la directory di lavoro corrente.

`rm`

Elimina file o directory.

`rmdir`

Elimina directory.

`touch`

Crea nuovi file vuoti o aggiorna il timestamp di modifica di un file esistente.

Risposte agli Esercizi Guidati

1. Tenendo presente quanto sotto riportato, seleziona le directory che verrebbero create con il comando `mkdir -p /tmp/outfiles/text/today /tmp/infiles/text/today`

```
$ pwd
/tmp
$ find
.
./outfiles
./outfiles/text
```

   Verranno create le directory contrassegnate. Le directory `/tmp`, `/tmp/outfiles` e `/tmp/outfiles/text` esistono già; quindi `mkdir` le ignorerà.

`/tmp`	
`/tmp/outfiles`	
`/tmp/outfiles/text`	
`/tmp/outfiles/text/today`	X
`/tmp/infiles`	X
`/tmp/infiles/text`	X
`/tmp/infiles/text/today`	X

2. Che cosa fa `-v` nei comandi `mkdir`, `rm` e `cp`?

   Normalmente, `-v` attiva l'output dettagliato. Fa in modo che i rispettivi programmi visualizzino ciò che stanno facendo nel momento in cui lo fanno:

```
$ rm -v a b
removed 'a'
removed 'b'
$ mv -v a b
'a' -> 'b'
$ cp -v b c
'b' -> 'c'
```

3. Che cosa succede se si tenta accidentalmente di copiare tre file sulla stessa Command Line in un file già esistente anziché in una directory?

`cp` si rifiuterà di fare qualsiasi cosa e mostrerà un messaggio di errore:

```
$ touch a b c d
$ cp a b c d
cp: target 'd' is not a directory
```

4. Che cosa succede quando usi `mv` per spostare una directory dentro se stessa?

   Viene visualizzato un messaggio di errore che dice che `mv` non può farlo.

```
$ mv a a
mv: cannot move 'a' to a subdirectory of itself, 'a/a'
```

5. Come potresti eliminare tutti i file nella directory corrente che iniziano con `old`?

   Dovresti usare il glob `old*` con il comando `rm`:

```
$ rm old*
```

6. Quale dei seguenti file corrisponde a `log_[a-z]_201?_*_01.txt`?

`log_3_2017_Jan_01.txt`	
`log_+_2017_Feb_01.txt`	
`log_b_2007_Mar_01.txt`	
`log_f_201A_Wednesday_01.txt`	X

```
$ ls log_[a-z]_201?_*_01.txt
log_f_201A_Wednesday_01.txt
```

   `log_[a-z]` corrisponde a `log_` seguito da una qualsiasi lettera minuscola; quindi ha corrispondenza sia con `log_f_201A_Wednesday_01.txt` che con `log_b_2007_Mar_01.txt`. `_201?` corrisponde a un qualunque carattere singolo dopo `_201`; quindi ha corrispondenza solo con `log_f_201A_Wednesday_01.txt`. Infine `*_01.txt` corrisponde a tutto ciò che finisce con `_01.txt`; quindi ha corrispondenza con l'opzione che è rimasta.

7. Indica alcuni glob che abbiano corrispondenza con tutti i file in elenco:

```
doc100
```

```
doc200
doc301
doc401
```

Esistono diverse soluzioni. Eccone alcune:

```
doc*
doc[1-4]*
doc?0?
doc[1-4]0?
```

Risposte agli Esercizi Esplorativi

1. Usa la pagina man di `cp` per scoprire come fare la copia di un file mantenendo i permessi e la data di modifica del file originale.

   Dovresti usare l'opzione `-p`. Dalla pagina man:

   ```
   $ man cp
   -p     same as --preserve=mode,ownership,timestamps
   --preserve[=ATTR_LIST]
                preserve the specified attributes (default: mode,ownership,time-
                stamps), if  possible  additional  attributes:  context,  links,
                xattr, all
   ```

2. Che cosa fa il comando `rmdir -p`? Provalo e spiega come si differenzia da `rm -r`.

   Fa sì che `rmdir` si comporti in modo simile a `mkdir -p`. Se gli viene passato un albero di directory vuote, le rimuoverà tutte.

   ```
   $ find
   .
   ./a
   ./a/b
   ./a/b/c
   $ rmdir -p a/b/c
   $ ls
   ```

3. NON ESEGUIRE REALMENTE QUESTO COMANDO: Cosa pensi che farebbe `rm -ri /*`? (SUL SERIO, NON PROVARE A ESEGUIRLO!)

   Rimuoverà tutti i file e le directory scrivibili dal tuo account utente. Ciò include qualsiasi file system di rete.

4. Oltre a usare `-i`, è possibile impedire a `mv` di sovrascrivere i file di destinazione?

   Sì, l'opzione `-n` o `--no-clobber` impedisce a `mv` di sovrascrivere i file.

   ```
   $ cat a
   a
   $ cat b
   b
   ```

```
$ mv -n a b
$ cat b
b
```

5. Spiega il comando `cp -u`.

   L'opzione `-u` fa in modo che `cp` copi un file solo se la destinazione non esiste o se è più vecchia del file sorgente.

```
$ ls -l
total 24K
drwxr-xr-x 123 emma student  12K Feb  2 05:34 ..
drwxr-xr-x   2 emma student 4.0K Feb  2 06:56 .
-rw-r--r--   1 emma student    2 Feb  2 06:56 a
-rw-r--r--   1 emma student    2 Feb  2 07:00 b
$ cat a
a
$ cat b
b
$ cp -u a b
$ cat b
b
$ cp -u a c
$ ls -l
total 12
-rw-r--r-- 1 emma student 2 Feb  2 06:56 a
-rw-r--r-- 1 emma student 2 Feb  2 07:00 b
-rw-r--r-- 1 emma student 2 Feb  2 07:00 c
```

Linux Professional Institute

Argomento 3: Il Potere della Command Line

 | Questa opera è concessa in licenza CC BY-NC-ND 4.0. |

3.1 Archiviazione dei File sulla Command Line

Obiettivi LPI di riferimento

Linux Essentials version 1.6, Exam 010, Objective 3.1

Peso

2

Aree di Conoscenza Chiave

- File, directory
- Archivi, compressione

Di seguito è riportato un elenco parziale dei file, dei termini e dei comandi utilizzati

- `tar`
- Comuni opzioni di `tar`
- `gzip`, `bzip2`, `xz`
- `zip`, `unzip`

3.1 Lezione 1

Certificazione:	Linux Essentials
Versione:	1.6
Argomento:	3 Il Potere della Command Line
Obiettivo:	3.1 Archiviazione dei File sulla Command Line
Lezione:	1 di 1

Introduzione

La compressione viene utilizzata per ridurre la quantità di spazio consumata da un insieme specifico di dati. In generale, la compressione viene utilizzata per ridurre la quantità di spazio necessaria per memorizzare un file. Un altro uso comune consiste nel ridurre la quantità di dati inviati attraverso una connessione di rete.

Il principio alla base della compressione prevede la sostituzione di pattern ripetitivi presenti nei dati. Supponi di avere un romanzo: alcune parole sono estremamente comuni, ma sono costituite da più caratteri, come per esempio la parola “il”. Potresti ridurre le dimensioni del romanzo in modo significativo se sostituissi queste parole e questi pattern comuni costituiti da più caratteri con un singolo carattere. Per esempio, puoi sostituire “il” con una lettera greca che non è usata altrove nel testo. Gli algoritmi di compressione dei dati funzionano in modo simile, ma con una maggiore complessità.

La compressione prevede due varianti: *senza perdita di dati* (lossless) e *con perdita di dati* (lossy). Ciò che viene compresso con un algoritmo lossless può essere decompresso nella sua forma originale. I dati compressi con un algoritmo lossy non possono essere recuperati. Gli algoritmi lossy vengono spesso utilizzati per immagini, video e audio in cui la perdita di qualità è

impercettibile per gli esseri umani, irrilevante per il contesto, o quando la perdita vale lo spazio risparmiato o il *throughput* di rete.

Gli strumenti di archiviazione vengono utilizzati per raggruppare file e directory in un unico file. Alcuni usi comuni sono: i backup, il raggruppamento del codice sorgente del software e la conservazione dei dati.

Archiviazione e compressione di solito vanno di pari passo. Alcuni strumenti di archiviazione comprimono addirittura i loro contenuti per impostazione predefinita. Altri possono farlo in modo opzionale. Alcuni strumenti di archiviazione devono essere utilizzati insieme a strumenti di compressione indipendenti se si desidera comprimere il contenuto.

Lo strumento più comune per archiviare file su sistemi Linux è `tar`. La maggior parte delle distribuzioni Linux include la versione GNU di `tar` e per questo motivo sarà quella trattata in questa lezione. `tar` da solo gestisce solamente l'archiviazione dei file, ma non è in grado di comprimerli.

Su Linux sono disponibili molti strumenti di compressione. I più utilizzati senza perdita di dati sono: `bzip2`, `gzip` e `xz`. Tutti e tre sono presenti nella maggior parte dei sistemi. Su sistemi vecchi o minimali potresti però non trovare installato `xz` o `bzip`. Diventando un utente Linux abituale probabilmente incontrerai file compressi con tutti e tre questi strumenti di compressione. Questi tre strumenti utilizzano algoritmi di compressione differenti: quindi un file compresso con uno strumento non può essere decompresso con un altro. Gli strumenti di compressione si fondano su un compromesso. Se desideri un rapporto di compressione elevato, sarà necessario più tempo per comprimere e decomprimere il file. Questo perché una compressione più elevata richiede più lavoro per trovare pattern più complessi. Tutti questi strumenti comprimono i dati, ma non possono creare archivi contenenti più file.

Gli strumenti di compressione indipendenti non sono in genere disponibili sui sistemi Windows. Gli strumenti di compressione e archiviazione di Windows sono di solito uniti insieme. Tienilo a mente se hai sistemi Linux e Windows che devono condividere file.

I sistemi Linux hanno anche strumenti per la gestione dei file `.zip` comunemente usati sui sistemi Windows. Si chiamano `zip` e `unzip`. Questi strumenti non sono installati di default su tutti i sistemi; quindi, se ne hai bisogno, potrebbe essere necessario installarli. Fortunatamente sono in genere presenti nei repository dei pacchetti delle varie distribuzioni.

Strumenti di Compressione

La quantità di spazio su disco risparmiata comprimendo i file dipende da alcuni fattori: la natura dei dati che stai comprimendo, l'algoritmo utilizzato per comprimere i dati e il livello di

compressione applicato. Non tutti gli algoritmi supportano diversi livelli di compressione.

Cominciamo preparando alcuni file di test da comprimere:

```
$ mkdir ~/linux_essentials-3.1
$ cd ~/linux_essentials-3.1
$ mkdir compression archiving
$ cd compression
$ cat /etc/* > bigfile 2> /dev/null
```

Ora creiamo tre copie di questo file:

```
$ cp bigfile bigfile2
$ cp bigfile bigfile3
$ cp bigfile bigfile4
$ ls -lh
total 2.8M
-rw-r--r-- 1 emma emma 712K Jun 23 08:08 bigfile
-rw-r--r-- 1 emma emma 712K Jun 23 08:08 bigfile2
-rw-r--r-- 1 emma emma 712K Jun 23 08:08 bigfile3
-rw-r--r-- 1 emma emma 712K Jun 23 08:08 bigfile4
```

Ora comprimiamo i file con ciascuno degli strumenti di compressione menzionati in precedenza:

```
$ bzip2 bigfile2
$ gzip bigfile3
$ xz bigfile4
$ ls -lh
total 1.2M
-rw-r--r-- 1 emma emma 712K Jun 23 08:08 bigfile
-rw-r--r-- 1 emma emma 170K Jun 23 08:08 bigfile2.bz2
-rw-r--r-- 1 emma emma 179K Jun 23 08:08 bigfile3.gz
-rw-r--r-- 1 emma emma 144K Jun 23 08:08 bigfile4.xz
```

Confronta le dimensioni dei file compressi con il file non compresso chiamato `bigfile`. Nota anche come gli strumenti di compressione abbiano aggiunto delle estensioni ai nomi dei file e rimosso i file non compressi.

Usa `bunzip2`, `gunzip` o `unxz` per decomprimere i file:

```
$ bunzip2 bigfile2.bz2
```

```
$ gunzip bigfile3.gz
$ unxz bigfile4.xz
$ ls -lh
total 2.8M
-rw-r--r-- 1 emma emma 712K Jun 23 08:20 bigfile
-rw-r--r-- 1 emma emma 712K Jun 23 08:20 bigfile2
-rw-r--r-- 1 emma emma 712K Jun 23 08:20 bigfile3
-rw-r--r-- 1 emma emma 712K Jun 23 08:20 bigfile4
```

Nota ancora una volta che il file compresso viene eliminato dopo essere stato decompresso.

Alcuni strumenti consentono diversi livelli di compressione. Un livello di compressione più alto in genere richiede più memoria e cicli di CPU, ma si traduce in un file compresso più piccolo. Accade il contrario per un livello di compressione più basso. Di seguito è riportata una dimostrazione con `xz` e `gzip`:

```
$ cp bigfile bigfile-gz1
$ cp bigfile bigfile-gz9
$ gzip -1 bigfile-gz1
$ gzip -9 bigfile-gz9
$ cp bigfile bigfile-xz1
$ cp bigfile bigfile-xz9
$ xz -1 bigfile bigfile-xz1
$ xz -9 bigfile bigfile-xz9
$ ls -lh bigfile bigfile-* *
total 3.5M
-rw-r--r-- 1 emma emma 712K Jun 23 08:08 bigfile
-rw-r--r-- 1 emma emma 205K Jun 23 13:14 bigfile-gz1.gz
-rw-r--r-- 1 emma emma 178K Jun 23 13:14 bigfile-gz9.gz
-rw-r--r-- 1 emma emma 156K Jun 23 08:08 bigfile-xz1.xz
-rw-r--r-- 1 emma emma 143K Jun 23 08:08 bigfile-xz9.xz
```

Non è necessario decomprimere un file ogni volta che lo si utilizza. Gli strumenti di compressione in genere includono delle versioni speciali degli strumenti più comuni utilizzati per leggere i file di testo. Per esempio, `gzip` ha una versione di `cat`, `grep`, `diff`, `less`, `more` e alcuni altri. Per `gzip` tali strumenti sono preceduti da una `z`, mentre per `bzip2` il prefisso è `bz` e per `xz` il prefisso è `xz`. Di seguito è riportato un esempio di utilizzo di `zcat` per leggere un file compresso con `gzip`:

```
$ cp /etc/hosts ./
$ gzip hosts
$ zcat hosts.gz
127.0.0.1   localhost
```

```
# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Strumenti di Archiviazione

Il programma `tar` è probabilmente lo strumento di archiviazione più utilizzato sui sistemi Linux. Nel caso ti stia chiedendo perché si chiama così, è l'abbreviazione di "tape archive" (archivio su nastro). I file creati con `tar` sono spesso chiamati *tar ball*. È molto comune che le applicazioni distribuite come codice sorgente siano delle tar ball.

La versione GNU di `tar` inclusa nelle distribuzioni Linux ha molte opzioni. Questa lezione tratterà quelle più comunemente usate.

Cominciamo creando un archivio dei file utilizzati per la compressione:

```
$ cd ~/linux_essentials-3.1
$ tar cf archiving/3.1.tar compression
```

L'opzione `c` indica a `tar` di creare un nuovo file di archivio e l'opzione `f` specifica il nome del file da creare. L'argomento che segue immediatamente le opzioni sarà sempre il nome del file su cui lavorare. Gli argomenti rimanenti sono i percorsi di qualsiasi file o directory che si desidera aggiungere, elencare o estrarre dal file. Nell'esempio riportato, abbiamo aggiunto all'archivio la directory `compression` e tutto il suo contenuto.

Per visualizzare il contenuto di una tar ball, usa l'opzione `t` di `tar`:

```
$ tar -tf 3.1.tar
compression/
compression/bigfile-xz1.xz
compression/bigfile-gz9.gz
compression/hosts.gz
compression/bigfile2
compression/bigfile
compression/bigfile-gz1.gz
compression/bigfile-xz9.xz
compression/bigfile3
compression/bigfile4
```

Nota come le opzioni siano precedute da `-`. A differenza della maggior parte dei programmi, il carattere `-` non è richiesto con `tar` quando si specificano le opzioni, sebbene non causi alcun danno se viene utilizzato.

NOTE Puoi usare l'opzione `-v` per lasciare che `tar` mostri come output i nomi dei file su cui opera durante la creazione o l'estrazione di un archivio.

Ora estraiamo il file:

```
$ cd ~/linux_essentials-3.1/archiving
$ ls
3.1.tar
$ tar xf 3.1.tar
$ ls
3.1.tar  compression
```

Supponi di aver bisogno di un solo file dell'archivio. In tal caso è possibile specificarlo dopo il nome dell'archivio. È anche possibile specificare più file se necessario:

```
$ cd ~/linux_essentials-3.1/archiving
$ rm -rf compression
$ ls
3.1.tar
$ tar xvf 3.1.tar compression/hosts.gz
compression/
compression/bigfile-xz1.xz
compression/bigfile-gz9.gz
compression/hosts.gz
compression/bigfile2
compression/bigfile
compression/bigfile-gz1.gz
compression/bigfile-xz9.xz
compression/bigfile3
compression/bigfile4
$ ls
3.1.tar  compression
$ ls compression
hosts.gz
```

A eccezione dei percorsi assoluti (percorsi che iniziano con `/`), i file `tar` preservano l'intero percorso dei file quando vengono creati. Poiché il file `3.1.tar` è stato creato con un'unica directory, questa directory verrà creata, una volta estratta, in relazione alla directory di lavoro

corrente. Un altro esempio dovrebbe rendere tutto più chiaro:

```
$ cd ~/linux_essentials-3.1/archiving
$ rm -rf compression
$ cd ../compression
$ tar cf ../tar/3.1-nodir.tar *
$ cd ../archiving
$ mkdir untar
$ cd untar
$ tar -xf ../3.1-nodir.tar
$ ls
bigfile   bigfile3  bigfile-gz1.gz  bigfile-xz1.xz  hosts.gz
bigfile2  bigfile4  bigfile-gz9.gz  bigfile-xz9.xz
```

TIP Se desideri utilizzare il percorso assoluto in un file `tar`, devi usare l'opzione `P`. Tieni presente che questo potrebbe sovrascrivere file importanti e potrebbe causare errori nel tuo sistema.

Il programma `tar` può anche gestire sul momento la compressione e la decompressione degli archivi. Per fare questo `tar` chiama uno degli strumenti di compressione discussi in precedenza in questa lezione: basta aggiungere l'opzione corrispondente all'algoritmo di compressione. Quelle più comunemente usate sono `j`, `J` e `z` rispettivamente per `bzip2`, `xz` e `gzip`. Ecco alcuni esempi che utilizzano gli algoritmi sopra menzionati:

```
$ cd ~/linux_essentials-3.1/compression
$ ls
bigfile   bigfile3  bigfile-gz1.gz  bigfile-xz1.xz  hosts.gz
bigfile2  bigfile4  bigfile-gz9.gz  bigfile-xz9.xz
$ tar -czf gzip.tar.gz bigfile bigfile2 bigfile3
$ tar -cjf bzip2.tar.bz2 bigfile bigfile2 bigfile3
$ tar -cJf xz.tar.xz bigfile bigfile2 bigfile3
$ ls -l | grep tar
-rw-r--r-- 1 emma emma  450202 Jun 27 05:56 bzip2.tar.bz2
-rw-r--r-- 1 emma emma  548656 Jun 27 05:55 gzip.tar.gz
-rw-r--r-- 1 emma emma  147068 Jun 27 05:56 xz.tar.xz
```

Nota come nell'esempio i file `.tar` abbiano dimensioni diverse. Questo mostra che sono stati compressi con successo. Se crei archivi `.tar` compressi dovresti sempre aggiungere una seconda estensione che indica l'algoritmo utilizzato. Queste sono `.xz`, `.bz` e `.gz` rispettivamente per `xz`, `bzip2` e `gzip`. A volte vengono utilizzate estensioni abbreviate, come per esempio `.tgz`.

È possibile aggiungere file ad archivi tar non compressi già esistenti usando l'opzione `u`. Se tenti di aggiungere un file a un archivio compresso, riceverai un errore.

```
$ cd ~/linux_essentials-3.1/compression
$ ls
bigfile   bigfile3  bigfile-gz1.gz  bigfile-xz1.xz  bzip2.tar.bz2  hosts.gz
bigfile2  bigfile4  bigfile-gz9.gz  bigfile-xz9.xz  gzip.tar.gz    xz.tar.xz
$ tar cf plain.tar bigfile bigfile2 bigfile3
$ tar tf plain.tar
bigfile
bigfile2
bigfile3
$ tar uf plain.tar bigfile4
$ tar tf plain.tar
bigfile
bigfile2
bigfile3
bigfile4
$ tar uzf gzip.tar.gz bigfile4
tar: Cannot update compressed archives
Try 'tar --help' or 'tar --usage' for more information.
```

Gestire i File ZIP

Le macchine Windows spesso non hanno applicazioni per gestire tar ball o molti degli strumenti di compressione che si trovano comunemente sui sistemi Linux. Se devi interagire con sistemi Windows, puoi utilizzare i file ZIP. Un file ZIP è un file di archivio simile a un file `tar` compresso.

I programmi `zip` e `unzip` possono essere usati per lavorare con i file ZIP su sistemi Linux. L'esempio seguente dovrebbe mostrare tutto ciò di cui hai bisogno per iniziare a utilizzarli. Per prima cosa creiamo una serie di file:

```
$ cd ~/linux_essentials-3.1
$ mkdir zip
$ cd zip/
$ mkdir dir
$ touch dir/file1 dir/file2
```

Ora usiamo `zip` per impachettare questi file in un file ZIP:

```
$ zip -r zipfile.zip dir
```

```
  adding: dir/ (stored 0%)
  adding: dir/file1 (stored 0%)
  adding: dir/file2 (stored 0%)
$ rm -rf dir
```

Infine, scompattiamo di nuovo il file ZIP:

```
$ ls
zipfile.zip
$ unzip zipfile.zip
Archive:  zipfile.zip
   creating: dir/
 extracting: dir/file1
 extracting: dir/file2
$ find
.
./zipfile.zip
./dir
./dir/file1
./dir/file2
```

Quando si aggiungono le directory ai file ZIP, l'opzione `-r` fa in modo che `zip` includa il contenuto di quelle directory. Senza questa opzione, avresti una directory vuota nel file ZIP.

Esercizi Guidati

1. Basandoti sulle estensioni, quali dei seguenti strumenti sono stati utilizzati per creare questi file?

Nome del File	`tar`	`gzip`	`bzip2`	`xz`
`archive.tar`				
`archive.tgz`				
`archive.tar.xz`				

2. Basandoti sulle estensioni, quali di questi file sono archivi e quali sono compressi?

Nome del File	**Archivio**	**Compresso**
`file.tar`		
`file.tar.bz2`		
`file.zip`		
`file.xz`		

3. Come potresti aggiungere un file a un file `tar` compresso con `gzip`?

4. Quale opzione di `tar` indica a `tar` di includere il carattere iniziale `/` nei percorsi assoluti?

5. `zip` supporta diversi livelli di compressione?

Esercizi Esplorativi

1. `tar` supporta l'utilizzo dei glob nell'elenco dei file durante l'operazione di estrazione?

2. Come puoi assicurarti che un file decompresso sia identico al file prima che fosse compresso?

3. Cosa succede se provi a estrarre da un archivio `tar` un file che esiste già sul tuo filesystem?

4. Come potresti estrarre il file `archive.tgz` senza usare l'opzione `z` di `tar`?

Sommario

I sistemi Linux dispongono di numerosi strumenti di compressione e archiviazione. Questa lezione ha trattato i più comuni. Lo strumento di archiviazione più comune è `tar`. Se è necessario interagire con i sistemi Windows, `zip` e `unzip` possono creare ed estrarre file ZIP.

Il comando `tar` ha alcune opzioni che vale la pena memorizzare. Sono `x` per estrarre, `c` per creare, `t` per visualizzare il contenuto e `u` per aggiungere o sostituire file. L'opzione `v` elenca i file che vengono elaborati da `tar` durante la creazione o l'estrazione di un archivio.

Il tipico repository di una distribuzione Linux include molti strumenti di compressione. I più comuni sono `gzip`, `bzip2` e `xz`. Gli algoritmi di compressione generalmente supportano diversi livelli di compressione che permettono di ottimizzare la velocità o le dimensioni del file. I file possono essere decompressi con `gunzip`, `bunzip2` e `unxz`.

Gli strumenti di compressione includono normalmente dei programmi che si comportano come i comuni strumenti per i file di testo, con la differenza che funzionano su file compressi. Alcuni di loro sono `zcat`, `bzcat` e `xzcat`. Gli strumenti di compressione normalmente includono programmi con le funzionalità di `grep`, `more`, `less`, `diff` e `cmp`.

Comandi utilizzati negli esercizi:

`bunzip2`
: Decomprime un file compresso con `bzip2`.

`bzcat`
: Visualizza il contenuto di un file compresso con `bzip2`.

`bzip2`
: Comprime i file utilizzando l'algoritmo e il formato `bzip2`.

`gunzip`
: Decomprime un file compresso con `gzip`.

`gzip`
: Comprime i file utilizzando l'algoritmo e il formato `gzip`.

`tar`
: Crea, aggiorna, elenca ed estrae archivi `tar`.

`unxz`

Decomprime un file compresso con `xz`.

`unzip`

Decomprime ed estrae il contenuto da un file ZIP.

`xz` Comprime i file utilizzando l'algoritmo e il formato `xz`.

`zcat`

Visualizza il contenuto di un file compresso con `gzip`.

`zip`

Crea e comprime archivi ZIP.

Risposte agli Esercizi Guidati

1. Basandoti sulle estensioni, quali dei seguenti strumenti sono stati utilizzati per creare questi file?

Nome del File	`tar`	`gzip`	`bzip2`	`xz`
`archive.tar`	X			
`archive.tgz`	X	X		
`archive.tar.xz`	X			X

2. Basandoti sulle estensioni, quali di questi file sono archivi e quali sono compressi?

Nome del File	**Archivio**	**Compresso**
`file.tar`	X	
`file.tar.bz2`	X	X
`file.zip`	X	X
`file.xz`		X

3. Come potresti aggiungere un file a un file `tar` compresso con `gzip`?

   Dovresti decomprimere il file con `gunzip`, aggiungere il file con `tar uf` e quindi comprimerlo con `gzip`.

4. Quale opzione di `tar` indica a `tar` di includere il carattere iniziale `/` nei percorsi assoluti?

   L'opzione `-P`. Dalla pagina man:

```
-P, --absolute-names
       Don't strip leading slashes from file names when creating  archives
```

5. `zip` supporta diversi livelli di compressione?

   Sì. Dovresti usare `-#`, sostituendo `#` con un numero da 0 a 9. Dalla pagina man:

```
-#
(-0, -1, -2, -3, -4, -5, -6, -7, -8, -9)
       Regulate the speed of compression using the specified  digit  #,
       where  -0  indicates  no compression (store all files), -1 indi-
```

```
       cates the fastest compression speed (less  compression)  and  -9
       indicates  the  slowest  compression speed (optimal compression,
       ignores the suffix list). The default compression level is -6.

       Though still being worked, the intention is  this  setting  will
       control  compression  speed  for  all compression methods.  Cur-
       rently only deflation is controlled.
```

Risposte agli Esercizi Esplorativi

1. `tar` supporta l'utilizzo dei glob nell'elenco dei file durante l'operazione di estrazione?

   Sì, dovresti usare l'opzione `--wildcards`. Se si usano le opzioni senza trattino, `--wildcards` deve essere posizionato subito dopo il file `tar`. Per esempio:

   ```
   $ tar xf tarfile.tar --wildcards dir/file*
   $ tar --wildcards -xf tarfile.tar dir/file*
   ```

2. Come puoi assicurarti che un file decompresso sia identico al file prima che fosse compresso?

   Non devi fare nulla con gli strumenti trattati in questa lezione. Tutti e tre includono un checksum nel loro formato di file che viene verificato durante la decompressione.

3. Cosa succede se provi a estrarre da un archivio `tar` un file che esiste già sul tuo filesystem?

   Il file sul tuo filesystem viene sovrascritto con la versione che si trova nel file `tar`.

4. Come potresti estrarre il file `archive.tgz` senza usare l'opzione `z` di `tar`?

   Dovresti prima decomprimerlo con `gunzip`.

   ```
   $ gunzip archive.tgz
   $ tar xf archive.tar
   ```

3.2 Ricerca ed Estrazione di Dati dai File

Obiettivi LPI di riferimento

Linux Essentials version 1.6, Exam 010, Objective 3.2

Peso

3

Aree di Conoscenza Chiave

- Le pipe sulla command line
- Reindirizzamento I/O
- Espressioni regolari di base attraverso l'utilizzo di `.`, `[ ]`, `*`, e `?`

Di seguito è riportato un elenco parziale dei file, dei termini e dei comandi utilizzati

- `grep`
- `less`
- `cat`, `head`, `tail`
- `sort`
- `cut`
- `wc`

3.2 Lezione 1

Certificazione:	Linux Essentials
Versione:	1.6
Argomento:	3 Il Potere della Command Line
Obiettivo:	3.2 Ricerca ed Estrazione di Dati dai File
Lezione:	1 di 2

Introduzione

In questa lezione ci concentreremo su come redirigere o trasmettere informazioni da una sorgente all'altra con l'aiuto di specifici strumenti. La Command Line di Linux redirige le informazioni tramite specifici canali standard. La tastiera è considerata lo standard input (*stdin* o canale 0) di un comando e lo schermo è considerato lo standard output (*stdout* o canale 1). Esiste anche un altro canale che ha lo scopo di redirigere l'output di errore (*stderr* o canale 2) di un comando o i messaggi di errore di un programma. L'input e/o l'output possono essere rediretti.

Quando si esegue un comando, a volte si desidera passare determinate informazioni al comando stesso o redirigere l'output verso uno specifico file. Ciascuna di queste funzionalità verrà discussa in queste due lezioni.

Redirezione di I/O

La redirezione di I/O consente all'utente di redirigere le informazioni da o verso un comando utilizzando un file di testo. Come descritto in precedenza, lo standard input, lo standard output e lo standard error possono essere rediretti e le informazioni possono essere prese da file di testo.

Redirigere lo Standard Output

Per redirigere lo standard output su un file, invece che sullo schermo, dobbiamo utilizzare l'operatore `>` seguito dal nome del file. Se il file non esiste, ne verrà creato uno nuovo, altrimenti il file esistente verrà sovrascritto.

Per vedere il contenuto del file appena creato, possiamo utilizzare il comando `cat`. Per impostazione predefinita, questo comando visualizza il contenuto di un file sullo schermo. Consulta la pagina di manuale per saperne di più sulle sue funzionalità.

L'esempio seguente mostra come funziona l'operatore. Nel primo caso viene creato un nuovo file contenente il testo "Hello World!":

```
$ echo "Hello World!" > text
$ cat text
Hello World!
```

Nel secondo caso, lo stesso file viene sovrascritto con il nuovo testo:

```
$ echo "Hello!" > text
$ cat text
Hello!
```

Se vogliamo aggiungere nuove informazioni alla fine del file, dobbiamo utilizzare l'operatore `>>`. Questo operatore crea anche un nuovo file se non riesce a trovarne uno esistente.

Il primo esempio mostra l'aggiunta di nuovo testo. Come si può vedere, il nuovo testo è stato aggiunto nella riga successiva:

```
$ echo "Hello to you too!" >> text
$ cat text
Hello!
Hello to you too!
```

Il secondo esempio mostra la creazione di un nuovo file:

```
$ echo "Hello to you too!" >> text2
$ cat text2
Hello to you too!
```

Redirigere lo Standard Error

Per redirigere soltanto i messaggi di errore, un utente dovrà utilizzare l'operatore `2>` seguito dal nome del file in cui verranno scritti gli errori. Se il file non esiste, ne verrà creato uno nuovo, altrimenti il file verrà sovrascritto.

Come già spiegato, il canale per redirigere lo standard error è il *canale 2*. Quando si redirige lo standard error, il canale deve essere specificato, contrariamente a quanto accade per lo standard output poichè il *canale 1* è impostato di default. Per esempio, il seguente comando cerca un file o una directory chiamata `games` e scrive soltanto l'errore nel file `text-error`, visualizzando lo standard output sullo schermo:

```
$ find /usr games 2> text-error
/usr
/usr/share
/usr/share/misc
---------Omitted output----------
/usr/lib/libmagic.so.1.0.0
/usr/lib/libdns.so.81
/usr/games
$ cat text-error
find: `games': No such file or directory
```

NOTE Per maggiori informazioni sul comando `find`, consulta la sua pagina man.

Per esempio, il seguente comando viene eseguito senza errori e, pertanto, non verrà scritta alcuna informazione nel file `text-error`:

```
$ sort /etc/passwd 2> text-error
$ cat text-error
```

Come lo standard output, anche lo standard error può essere aggiunto a un file tramite l'operatore `2>>`, che aggiunge il nuovo errore alla fine del file. Se il file non esiste, ne verrà creato uno nuovo. Il primo esempio mostra l'aggiunta di nuove informazioni al file, mentre il secondo esempio mostra come il comando crei un nuovo file nel caso in cui non sia possibile trovarne uno esistente con lo stesso nome:

```
$ sort /etc 2>> text-error
$ cat text-error
sort: read failed: /etc: Is a directory
```

```
$ sort /etc/shadow 2>> text-error2
$ cat text-error2
sort: open failed: /etc/shadow: Permission denied
```

Con questo tipo di redirezione, soltanto i messaggi di errore verranno rediretti verso il file; l'output normale verrà scritto sullo schermo o passerà attraverso lo standard output o *stdout*.

Esiste un file particolare che tecnicamente è un *cestino per dati*, in inglese *bit bucket*, (un file che accetta dell'input, ma non fa nulla con esso): `/dev/null`. Puoi redirigere verso di esso tutte le informazioni non importanti che potresti non voler visualizzare o non voler redirigere verso un file importante. Un esempio è mostrato qui sotto:

```
$ sort /etc 2> /dev/null
```

Redirigere lo Standard Input

Questa tipologia di redirezione viene utilizzata per inviare dati in input a un comando da un file specificato anziché da tastiera. In questo caso viene utilizzato l'operatore `<`, come mostrato nell'esempio seguente:

```
$ cat < text
Hello!
Hello to you too!
```

La redirezione dello standard input viene solitamente utilizzata con i comandi che non accettano argomenti. Il comando `tr` è uno di questi. Questo comando può essere utilizzato per trasformare il contenuto di un file modificando i caratteri presenti in esso in un certo modo, per esempio eliminando un carattere specifico dal file. L'esempio seguente mostra come eliminare il carattere `l`:

```
$ tr -d "l" < text
Heo!
Heo to you too!
```

Per maggiori informazioni, consulta la pagina man di `tr`.

Here Document

A differenza della redirezione dell'output, l'operatore `<<` si comporta in modo diverso rispetto agli altri operatori. Questo flusso in ingresso è anche chiamato *here document*, che rappresenta il blocco di codice o di testo che può essere rediretto verso il comando o verso il programma interattivo. Diversi tipi di linguaggi di scripting, come `bash`, `sh` e `csh` sono in grado di ricevere dell'input direttamente dalla Command Line, senza utilizzare alcun file di testo.

Come si può vedere nell'esempio sotto riportato, l'operatore viene utilizzato per inviare dei dati al comando, mentre la parola che segue non specifica il nome del file. La parola viene interpretata come un delimitatore dell'input e non viene presa in considerazione come contenuto; quindi non verrà visualizzata da `cat`:

```
$ cat << hello
> hey
> ola
> hello
hey
ola
```

Consulta la pagina man del comando `cat` per avere maggiori informazioni.

Combinazioni

La prima combinazione di cui ci occuperemo combina la redirezione dello standard output e dello standard error verso lo stesso file. Vengono utilizzati gli operatori `&>` e `&>>`: `&` rappresenta la combinazione del *canale 1* e del *canale 2*. Il primo operatore sovrascrive il contenuto esistente del file, mentre il secondo accoda o aggiunge le nuove informazioni alla fine del file. Entrambi gli operatori consentono la creazione di un nuovo file nel caso in cui questo non esista, proprio come nelle sezioni precedenti:

```
$ find /usr admin &> newfile
$ cat newfile
/usr
/usr/share
/usr/share/misc
---------Omitted output----------
/usr/lib/libmagic.so.1.0.0
/usr/lib/libdns.so.81
/usr/games
find: `admin': No such file or directory
```

```
$ find /etc/calendar &>> newfile
$ cat newfile
/usr
/usr/share
/usr/share/misc
---------Omitted output----------
/usr/lib/libmagic.so.1.0.0
/usr/lib/libdns.so.81
/usr/games
find: `admin': No such file or directory
/etc/calendar
/etc/calendar/default
```

Vediamo un esempio che utilizza il comando `cut`:

```
$ cut -f 3 -d "/" newfile
$ cat newfile

share
share
share
---------Omitted output----------
lib
games
find: `admin': No such file or directory
calendar
calendar
find: `admin': No such file or directory
```

Il comando `cut`, con l'opzione `-f`, taglia i campi specificati dal file in ingresso, nel nostro caso il terzo campo. Affinché il comando trovi il campo, è necessario specificare anche un delimitatore con l'opzione `-d`. Nel nostro caso il delimitatore è il carattere `/`.

Per saperne di più sul comando `cut`, consulta la sua pagina man.

Pipe

La redirezione viene principalmente utilizzata per memorizzare il risultato di un comando che deve essere elaborato da un comando diverso. Questo tipo di processo intermedio può diventare molto noioso e complicato se si desidera che i dati passino attraverso più processi. Per evitare questo, puoi collegare i comandi direttamente tramite le *pipe*. In altre parole, l'output del primo comando diventa automaticamente l'input del secondo comando. Questa connessione viene fatta

utilizzando l'operatore `|` (barra verticale):

```
$ cat /etc/passwd | less
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
:
```

Nell'esempio sopra riportato il comando `less`, che si trova dopo l'operatore *pipe*, modifica il modo in cui il file viene visualizzato. Il comando `less` mostra il file di testo consentendo all'utente di scorrere su e giù di una riga per volta. `less` è anche usato di default per visualizzare le pagine man, come discusso nelle lezioni precedenti.

È possibile utilizzare più *pipe* contemporaneamente. I comandi intermedi che ricevono un input e che poi lo elaborano e producono un output sono chiamati *filtri*. Prendiamo il comando `ls -l` e proviamo a contare il numero di parole delle prime 10 righe in output. Per fare ciò, dovremo utilizzare il comando `head` che, per impostazione predefinita, mostra le prime 10 righe di un file e poi dovremo contare le parole usando il comando `wc`:

```
$ ls -l | head | wc -w
10
```

Come accennato in precedenza, per impostazione predefinita, `head` mostra solo le prime 10 righe del file di testo specificato. Questo comportamento può essere modificato utilizzando specifiche opzioni. Controlla la pagina man del comando per avere ulteriori informazioni.

Esiste un altro comando che visualizza la parte finale di un file: `tail`. Per impostazione predefinita, questo comando seleziona le ultime 10 righe e le visualizza, ma, come per `head`, tale numero può anche essere modificato. Controlla la pagina man di `tail` per maggiori dettagli.

NOTE L'opzione `-f` mostra le ultime righe di un file mentre viene aggiornato. Questa caratteristica può diventare molto utile quando si monitora un file come `syslog` durante lo svolgimento di una attività.

Il comando `wc` (acronimo di *word count*) conta, per impostazione predefinita, le righe, le parole e i byte di un file. Come mostrato nell'esempio sopra riportato l'opzione `-w` fa sì che il comando conti solamente le parole all'interno delle righe selezionate. Le opzioni più comuni di questo comando sono: `-l`, che indica al comando di contare solo le righe; `-c`, che viene utilizzato per contare solo i byte. Nella pagina man di questo comando puoi trovare ulteriori varianti e opzioni e maggiori informazioni su `wc`.

Esercizi Guidati

1. Elenca il contenuto della tua directory corrente, mostrando anche ownership e permessi, e redirigi l'output su un file chiamato `contents.txt` all'interno della tua directory home.

2. Ordina il contenuto del file `contents.txt` dalla directory corrente e aggiungilo alla fine di un nuovo file chiamato `contents-sorted.txt`.

3. Mostra le ultime 10 righe del file `/etc/passwd` e redirigile su un nuovo file nella directory `Documents` del tuo utente.

4. Conta il numero di parole all'interno del file `contents.txt` e aggiungi l'output alla fine del file `field2.txt` nella tua directory home. Dovrai utilizzare sia la redirezione dell'input che dell'output.

5. Mostra le prime 5 righe del file `/etc/passwd` e ordina l'output in ordine alfabetico inverso.

6. Utilizzando il file `contents.txt` creato in precedenza, conta il numero di caratteri delle ultime 9 righe.

7. Conta il numero di file con nome `test` all'interno della directory `/usr/share` e delle sue sottodirectory. Nota: ogni riga di output del comando `find` rappresenta un file.

Esercizi Esplorativi

1. Seleziona il secondo campo del file `contents.txt` e redirigi lo standard output e lo standard error su un altro file chiamato `field1.txt`.

2. Utilizzando l'operatore di redirezione dell'input e il comando `tr`, elimina i trattini (`-`) dal file `contents.txt`.

3. Qual è il più grande vantaggio che si ha nel redirigere solo gli errori su un file?

4. Sostituisci con un singolo spazio tutti gli spazi ripetuti all'interno del file `contents.txt` ordinato alfabeticamente.

5. Da Command Line, elimina gli spazi ripetuti (come fatto nell'esercizio precedente), seleziona il nono campo e ordina in base a esso in ordine alfabetico inverso senza fare distinzione tra maiuscole e minuscole. Quante *pipe* hai dovuto usare?

Sommario

In questa lezione hai imparato:

- I tipi di redirezione;
- Come utilizzare gli operatori di redirezione;
- Come utilizzare le *pipe* per filtrare l'output dei comandi.

Comandi utilizzati in questa lezione:

`cut`
Rimuove sezioni da ogni riga di un file.

`cat`
Visualizza o concatena file.

`find`
Cerca i file in una gerarchia di directory.

`less`
Visualizza un file, consentendo all'utente di scorrere una riga alla volta.

`more`
Visualizza un file, una pagina alla volta.

`head`
Visualizza le prime 10 righe di un file.

`tail`
Visualizza le ultime 10 righe di un file.

`sort`
Ordina i file.

`wc`
Conta, per impostazione predefinita, le righe, le parole e i byte di un file.

Risposte agli Esercizi Guidati

1. Elenca il contenuto della tua directory corrente, mostrando anche ownership e permessi, e redirigi l'output su un file chiamato `contents.txt` all'interno della tua directory home.

```
$ ls -l > contents.txt
```

2. Ordina il contenuto del file `contents.txt` dalla directory corrente e aggiungilo alla fine di un nuovo file chiamato `contents-sorted.txt`.

```
$ sort contents.txt >> contents-sorted.txt
```

3. Mostra le ultime 10 righe del file `/etc/passwd` e redirigile su un nuovo file nella directory `Documents` del tuo utente.

```
$ tail /etc/passwd > Documents/newfile
```

4. Conta il numero di parole all'interno del file `contents.txt` e aggiungi l'output alla fine del file `field2.txt` all'interno della tua home directory. Sarà necessario utilizzare sia il reindirizzamento di input sia il reindirizzamento di output.

```
$ wc -w < contents.txt >> field2.txt
```

5. Mostra le prime 5 righe del file `/etc/passwd` e ordina l'output in ordine alfabetico inverso.

```
$ head -n 5 /etc/passwd | sort -r
```

6. Utilizzando il file `contents.txt` creato in precedenza, conta il numero di caratteri delle ultime 9 righe.

```
$ tail -n 9 contents.txt | wc -c
531
```

7. Conta il numero di file con nome `test` all'interno della directory `/usr/share` e delle sue sottodirectory. Nota: ogni riga di output del comando `find` rappresenta un file.

```
$ find /usr/share -name test | wc -l
125
```

Risposte agli Esercizi Esplorativi

1. Seleziona il secondo campo del file `contents.txt` e redirigi lo standard output e lo standard error su un altro file chiamato `field1.txt`.

   ```
   $ cut -f 2 -d " " contents.txt &> field1.txt
   ```

2. Utilizzando l'operatore di redirezione dell'input e il comando `tr`, elimina i trattini (`-`) dal file `contents.txt`.

   ```
   $ tr -d "-" < contents.txt
   ```

3. Qual è il più grande vantaggio che si ha nel redirigere solo gli errori su un file?

   Redirigere solamente gli errori su un file può aiutare a mantenere un file di log che viene monitorato di frequente.

4. Sostituisci con un singolo spazio tutti gli spazi ripetuti all'interno del file `contents.txt` ordinato alfabeticamente.

   ```
   $ sort contents.txt | tr -s " "
   ```

5. Da Command Line, elimina gli spazi ripetuti (come fatto nell'esercizio precedente), seleziona il nono campo e ordina in base a esso in ordine alfabetico inverso senza fare distinzione tra maiuscole e minuscole. Quante *pipe* hai dovuto usare?

   ```
   $ cat contents.txt | tr -s " " | cut -f 9 -d " " | sort -fr
   ```

   L'esercizio utilizza 3 *pipe*, una per ogni filtro.

Linux Professional Institute

3.2 Lezione 2

Certificazione:	Linux Essentials
Versione:	1.6
Argomento:	3 Il Potere della Command Line
Obiettivo:	3.2 Ricerca ed Estrazione di Dati dai File
Lezione:	2 di 2

Introduzione

In questa lezione esamineremo gli strumenti utilizzati per manipolare i testi. Questi strumenti vengono utilizzati frequentemente dagli amministratori di sistema o dai programmi per monitorare o identificare automaticamente informazioni ricorrenti specifiche.

Cercare all'interno dei File con `grep`

Il primo strumento di cui parleremo in questa lezione è il comando `grep`, acronimo di "global regular expression print"; la sua funzionalità principale è la ricerca di uno specifico pattern all'interno dei file. Il comando restituisce le righe contenenti il pattern specificato, evidenziato in rosso.

```
$ grep bash /etc/passwd
root:x:0:0:root:/root:/bin/bash
user:x:1001:1001:User,,,,:/home/user:/bin/bash
```

`grep`, come la maggior parte dei comandi, può anche modificare il suo comportamento base a

seconda delle opzioni utilizzate. Ecco le più comuni:

`-i`
: la ricerca non fa distinzione tra lettere maiuscole e minuscole

`-r`
: la ricerca è ricorsiva (cerca in tutti i file all'interno della directory specificata e delle sue sottodirectory)

`-c`
: la ricerca conta il numero di corrispondenze trovate

`-v`
: inverte la ricerca, restituendo le righe che non hanno corrispondenza con i termini di ricerca

`-E`
: attiva le espressioni regolari estese (necessarie per alcuni dei meta-caratteri più avanzati come `|` , `+` e `?`)

`grep` ha molte altre opzioni utili. Consulta la pagina man per saperne di più.

Espressioni Regolari

Il secondo strumento è molto potente. Viene utilizzato per descrivere frammenti di testo all'interno dei file, detti anche *espressioni regolari*. Le espressioni regolari sono estremamente utili per estrarre dati da file di testo attraverso la costruzione di pattern. Sono comunemente usati all'interno di script o nella programmazione con linguaggi di alto livello, come Perl o Python.

Quando si lavora con le espressioni regolari è molto importante tenere presente che *conta ogni carattere* e che il pattern viene scritto con lo scopo di trovare una corrispondenza con una specifica sequenza di caratteri, nota anche come *stringa*. La maggior parte dei pattern utilizza i normali simboli ASCII: lettere, cifre, punteggiatura o altri simboli, ma può anche utilizzare caratteri Unicode per indicare la corrispondenza con qualsiasi altro tipo di testo.

Il seguente elenco indica i meta-caratteri delle espressioni regolari utilizzati per formare i pattern.

`.`
: Corrisponde a qualsiasi singolo carattere (tranne il carattere di nuova riga)

`[abcABC]`
: Corrisponde a qualsiasi carattere all'interno delle parentesi

`[^abcABC]`

Corrisponde a qualsiasi carattere tranne quelli racchiusi tra parentesi

`[a-z]`

Corrisponde a qualsiasi carattere nell'intervallo indicato

`[^a-z]`

Corrisponde a qualsiasi carattere tranne quelli nell'intervallo indicato

`sun|moon`

Trova una qualsiasi delle stringhe indicate

`^`

Inizio di una riga

`$`

Fine di una riga

Tutte le funzionalità delle espressioni regolari possono essere implementate anche in `grep`. Puoi notare che nell'esempio precedente la parola non è racchiusa tra virgolette doppie. Per evitare che la shell interpreti i meta-caratteri, si consiglia di inserire i pattern più complessi tra virgolette doppie (" "). All'atto pratico, utilizzeremo le virgolette doppie durante l'implementazione delle espressioni regolari. Le altre virgolette mantengono la loro normale funzionalità, come discusso nelle lezioni precedenti.

I prossimi esempi evidenziano il funzionamento delle espressioni regolari. Avremo bisogno di dati all'interno del file: pertanto, la prossima serie di comandi aggiungerà semplicemente stringhe diverse al file `text.txt`.

```
$ echo "aaabbb1" > text.txt
$ echo "abab2" >> text.txt
$ echo "noone2" >> text.txt
$ echo "class1" >> text.txt
$ echo "alien2" >> text.txt
$ cat text.txt
aaabbb1
abab2
noone2
class1
alien2
```

Il primo esempio mostra una combinazione di ricerche all'interno del file senza e con le espressioni regolari. Per comprendere appieno le espressioni regolari, è molto importante mostrare la differenza tra le due ricerche. Il primo comando cerca la stringa esatta in un qualsiasi punto della riga, mentre il secondo comando cerca gli insiemi di caratteri che contengono uno qualsiasi dei caratteri tra parentesi. Pertanto i risultati dei comandi saranno diversi.

```
$ grep "ab" text.txt
aaabbb1
abab2
$ grep "[ab]" text.txt
aaabbb1
abab2
class1
alien2
```

La seconda serie di esempi mostra l'utilizzo dei meta-caratteri di inizio e fine riga. È molto importante sottolineare la necessità di posizionare i 2 caratteri al posto giusto all'interno dell'espressione. Quando si specifica l'inizio della riga, il meta-carattere deve essere posizionato prima dell'espressione, mentre quando si specifica la fine della riga, il meta-carattere deve essere posizionato dopo l'espressione.

```
$ grep "^a" text.txt
aaabbb1
abab2
alien2
$ grep "2$" text.txt
abab2
noone2
alien2
```

Oltre ai meta-caratteri spiegati sopra, le espressioni regolari hanno anche meta-caratteri che consentono la moltiplicazione del pattern precedente:

`*`

Zero o più occorrenze del pattern precedente

`+`

Una o più occorrenze del pattern precedente

`?`

Zero o una occorrenza del pattern precedente

In base ai meta-caratteri moltiplicatori, il comando seguente ricerca una stringa che contenga `ab`, un singolo carattere e uno o più dei caratteri trovati in precedenza. Il risultato mostra che `grep` ha trovato la stringa `aaabbb1` (la parte per la quale si ha corrispondenza è `abbb`) e la stringa `abab2`. Poiché il carattere `+` è un carattere delle espressioni regolari *estese*, dobbiamo passare l'opzione `-E` al comando `grep`.

```
$ grep -E "ab.+" text.txt
aaabbb1
abab2
```

Gran parte dei meta-caratteri sono autoesplicativi, ma all'inizio possono risultare un po' complicati. Gli esempi precedenti rappresentano una piccola parte delle funzionalità delle espressioni regolari: prova tutti i meta-caratteri della tabella sopra riportata per capire meglio come funzionano.

Esercizi Guidati

Utilizzando `grep` e il file `/usr/share/hunspell/en_US.dic`, trova le righe che corrispondono ai seguenti criteri:

1. Tutte le righe che contengono la parola `cat` in un qulunque punto della riga.

2. Tutte le righe che non contengono nessuno dei seguenti caratteri: `sawgtfixk`.

3. Tutte le righe che iniziano con 3 lettere qualsiasi e la parola `dig`.

4. Tutte le righe che finiscono con almeno una `e`.

5. Tutte le righe che contengono una delle seguenti parole: `org`, `kay` o `tuna`.

6. Il numero delle righe che iniziano con una o nessuna `c` seguita dalla stringa `ati`.

Esercizi Esplorativi

1. Individua l'espressione regolare che ha corrispondenza con le parole nella riga "Include" e non ha corrispondenza con quelle nella riga "Esclude":
   - Include: `pot`, `spot`, `apot`

     Esclude: `potic`, `spots`, `potatoe`

   - Include: `arp99`, `apple`, `zipper`

     Esclude: `zoo`, `arive`, `attack`

   - Include: `arcane`, `capper`, `zoology`

     Esclude: `air`, `coper`, `zoloc`

   - Include: `0th/pt`, `3th/tc`, `9th/pt`

     Esclude: `0/nm`, `3/nm`, `9/nm`

   - Include: `Hawaii`, `Dario`, `Ramiro`

     Esclude: `hawaii`, `Ian`, `Alice`

2. Quale altro utile comando è comunemente usato per cercare all'interno dei file? Quali funzionalità aggiuntive ha?
3. Scegli uno degli esempi della lezione precedente e, con l'aiuto di `grep`, prova a cercare un pattern specifico all'interno dell'output del comando.

Sommario

In questa lezione hai imparato:

- I meta-caratteri delle espressioni regolari;
- Come creare pattern con le espressioni regolari;
- Come cercare all'interno dei file.

Comandi utilizzati negli esercizi:

`grep`
Cerca caratteri o stringhe all'interno di un file

Risposte agli Esercizi Guidati

Utilizzando `grep` e il file `/usr/share/hunspell/en_US.dic`, trova le righe che corrispondono ai seguenti criteri:

1. Tutte le righe che contengono la parola `cat` in un qulunque punto della riga.

```
$ grep "cat" /usr/share/hunspell/en_US.dic
Alcatraz/M
Decatur/M
Hecate/M
...
```

2. Tutte le righe che non contengono nessuno dei seguenti caratteri: `sawgtfixk`.

```
$ grep -v "[sawgtfixk]" /usr/share/hunspell/en_US.dic
49269
0/nm
1/n1
2/nm
2nd/p
3/nm
3rd/p
4/nm
5/nm
6/nm
7/nm
8/nm
...
```

3. Tutte le righe che iniziano con 3 lettere qualsiasi e la parola `dig`.

```
$ grep "^...dig" /usr/share/hunspell/en_US.dic
cardigan/SM
condign
predigest/GDS
...
```

4. Tutte le righe che finiscono con almeno una `e`.

```
$ grep -E "e+$" /usr/share/hunspell/en_US.dic
Anglicize
Anglophobe
Anthropocene
...
```

5. Tutte le righe che contengono una delle seguenti parole: `org`, `kay` o `tuna`.

```
$ grep -E "org|kay|tuna" /usr/share/hunspell/en_US.dic
Borg/SM
George/MS
Tokay/M
fortunate/UY
...
```

6. Il numero delle righe che iniziano con una o nessuna `c` seguita dalla stringa `ati`.

```
$ grep -cE "^c?ati" /usr/share/hunspell/en_US.dic
3
```

Risposte agli Esercizi Esplorativi

1. Individua l'espressione regolare che ha corrispondenza con le parole nella riga "Include" e non ha corrispondenza con quelle nella riga "Esclude":
   - Include: `pot`, `spot`, `apot`

     Esclude: `potic`, `spots`, `potatoe`

     Soluzione: `pot$`
   - Include: `arp99`, `apple`, `zipper`

     Esclude: `zoo`, `arive`, `attack`

     Soluzione: `p+`
   - Include: `arcane`, `capper`, `zoology`

     Esclude: `air`, `coper`, `zoloc`

     Soluzione: `arc|cap|zoo`
   - Include: `0th/pt`, `3th/tc`, `9th/pt`

     Esclude: `0/nm`, `3/nm`, `9/nm`

     Soluzione: `[0-9]th.+`
   - Include: `Hawaii`, `Dario`, `Ramiro`

     Esclude: `hawaii`, `Ian`, `Alice`

     Soluzione: `^[A-Z]a.*i+`
2. Quale altro utile comando è comunemente usato per cercare all'interno dei file? Quali funzionalità aggiuntive ha?

   Il comando `sed`. Questo comando può trovare e sostituire caratteri o set di caratteri all'interno di un file.
3. Scegli uno degli esempi della lezione precedente e, con l'aiuto di `grep`, prova a cercare un pattern specifico all'interno dell'output del comando.

   Attenzione: ho scelto una delle risposte degli Esercizi Esplorativi e ho cercato la riga che ha i

permessi di lettura, scrittura ed esecuzione per il gruppo. La tua risposta potrebbe essere diversa, a seconda del comando che hai scelto e del pattern che hai creato.

```
$ cat contents.txt | tr -s " " | grep "^....rwx"
```

Questo esercizio serve per mostrarti che `grep` può anche ricevere un input da diversi comandi e può essere utile per filtrare le informazioni generate.

3.3 Trasformare i Comandi in uno Script

Obiettivi LPI di riferimento

Linux Essentials version 1.6, Exam 010, Objective 3.3

Peso

4

Aree di Conoscenza Chiave

- Shell scripting di base
- Conoscenza dei più comuni editor di testo (vi e nano)

Di seguito è riportato un elenco parziale dei file, dei termini e dei comandi utilizzati

- `#!` (shebang)
- `/bin/bash`
- Variabili
- Argomenti
- Cicli `for`
- `echo`
- Exit status

3.3 Lezione 1

Certificazione:	Linux Essentials
Versione:	1.6
Argomento:	3 Il Potere della Command Line
Obiettivo:	3.3 Trasformare i Comandi in uno Script
Lezione:	1 di 2

Introduzione

Finora abbiamo imparato a eseguire i comandi dalla shell, ma possiamo anche inserirli in un file e poi renderlo eseguibile. Quando viene lanciato il file, questi comandi vengono eseguiti in sequenza uno dopo l'altro. Questi file eseguibili sono chiamati *script* e sono uno strumento assolutamente indispensabile per qualsiasi amministratore di sistema Linux. In sostanza, possiamo considerare Bash un linguaggio di programmazione oltre che una shell.

Stampare l'output

Iniziamo mostrando un comando che potresti aver visto nelle lezioni precedenti: `echo` stampa un argomento sullo standard output.

```
$ echo "Hello World!"
Hello World!
```

Useremo ora la redirezione dei file per inviare questo comando a un nuovo file chiamato `new_script`.

```
$ echo 'echo "Hello World!"' > new_script
$ cat new_script
echo "Hello World!"
```

Il file `new_script` ora contiene lo stesso comando di prima.

Rendere uno Script Eseguibile

Mostriamo alcuni dei passaggi necessari per far sì che questo file funzioni nel modo previsto. La prima idea di un utente potrebbe essere quella di digitare semplicemente il nome dello script, proprio come digiterebbe il nome di un qualsiasi altro comando:

```
$ new_script
/bin/bash: new_script: command not found
```

Possiamo tranquillamente supporre che `new_script` esista nella nostra posizione corrente; nota però che il messaggio di errore non ci dice che il *file* non esiste, ma che il *comando* non esiste. Potrebbe essere utile discutere di come Linux gestisce i comandi e gli eseguibili.

Comandi e `PATH`

Quando per esempio digitiamo il comando `ls` nella shell, eseguiamo un file chiamato `ls` che esiste nel nostro filesystem. Per dimostrarlo, si può usare `which`:

```
$ which ls
/bin/ls
```

Diventerebbe subito noioso digitare il percorso assoluto di `ls` ogni volta che si desidera esaminare il contenuto di una directory: per questo motivo Bash ha una *variabile d'ambiente* che contiene tutte le directory in cui potremmo trovare i comandi che desideriamo eseguire. Puoi visualizzare il contenuto di questa variabile usando `echo`.

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/sn
ap/bin
```

In ciascuna di queste posizioni, separate tra loro dal carattere due punti (`:`), la shell si aspetta di trovare i comandi. Avrai notato che `/bin` è presente, ma è lecito ritenere che la nostra posizione

corrente non lo sia. La shell cercherà `new_script` in ciascuna di queste directory, ma non lo troverà e quindi mostrerà il messaggio di errore visto in precedenza.

Ci sono tre soluzioni a questo problema: possiamo spostare `new_script` in una delle directory presenti nella variabile `PATH`; possiamo aggiungere la nostra directory corrente a quelle presenti nella variabile `PATH`; oppure possiamo cambiare il modo in cui chiamiamo lo script. L'ultima soluzione è la più semplice: richiede semplicemente di specificare la *posizione corrente* quando si chiama lo script, utilizzando una barra obliqua dopo un punto (`./`).

```
$ ./new_script
/bin/bash: ./new_script: Permission denied
```

Il messaggio di errore è cambiato, il che indica che abbiamo fatto dei progressi.

Permessi di Esecuzione

La prima indagine che un utente dovrebbe fare in questo caso consiste nel dare un'occhiata al file tramite `ls -l`:

```
$ ls -l new_script
-rw-rw-r-- 1 user user 20 Apr 30 12:12 new_script
```

Possiamo vedere che i permessi di questo file sono impostati di default su `664`. Non abbiamo infatti ancora impostato i *permessi di esecuzione*.

```
$ chmod +x new_script
$ ls -l new_script
-rwxrwxr-x 1 user user 20 Apr 30 12:12 new_script
```

Questo comando imposta i permessi di esecuzione per *tutti* gli utenti. Tieni presente che questo potrebbe comportare un rischio per la sicurezza, ma per il momento rappresenta un livello di permessi accettabile.

```
$ ./new_script
Hello World!
```

Ora siamo in grado di eseguire il nostro script.

Definire un Interprete

Come abbiamo mostrato è possibile inserire semplicemente del testo in un file, impostarlo come eseguibile ed eseguirlo. `new_script` è ancora tecnicamente un normale file di testo, ma siamo riusciti a fare in modo che venisse interpretato da Bash. Ma cosa succederebbe se fosse scritto in Perl o in Python?

È buona norma specificare il *tipo di interprete* che si desidera utilizzare nella prima riga di uno script. Questa riga è chiamata *bang line* o più comunemente *shebang* e indica al sistema come vogliamo eseguire il file. Dato che stiamo imparando Bash, useremo il percorso assoluto del nostro eseguibile Bash, trovandolo ancora una volta tramite `which`:

```
$ which bash
/bin/bash
```

Il nostro shebang inizia con un segno cancelletto e un punto esclamativo, seguito dal percorso assoluto individuato in precedenza. Apriamo `new_script` con un editor di testo e inseriamo lo shebang. Cogliamo anche l'occasione per inserire un *commento* nel nostro script. I commenti vengono ignorati dall'interprete. Sono scritti per gli altri utenti che desiderano comprendere lo script.

```
#!/bin/bash

# This is our first comment. It is also good practice to document all scripts.

echo "Hello World!"
```

Cambiamo anche il nome del file salvandolo come `new_script.sh`. Il suffisso `.sh` non modifica in alcun modo l'esecuzione del file. È convenzione attribuire agli script bash le estensioni `.sh` o `.bash` per identificarli più facilmente, proprio come gli script Python vengono solitamente identificati dal suffisso `.py`.

Editor di Testo Comuni

Gli utenti Linux spesso devono lavorare in un ambiente in cui non sono disponibili degli editor di testo grafici. Per tale motivo si consiglia vivamente di sviluppare almeno una certa dimestichezza nel modificare i file di testo da Command Line. Due dei più comuni editor di testo sono `vi` e `nano`.

vi

`vi` è un antico editor di testo ed è installato di default su quasi tutti i sistemi Linux esistenti. Da `vi` ha avuto origine un clone chiamato *vi IMproved* o `vim` che aggiunge alcune funzionalità, mantenendo però l'interfaccia di `vi`. Anche se lavorare con `vi` può sembrare difficoltoso per un nuovo utente, questo editor è molto popolare e amato da coloro che ne conoscono le numerose funzionalità.

La differenza più importante tra `vi` e applicazioni quali per esempio, il Blocco Note è che `vi` ha tre diverse modalità. All'avvio, i tasti `H`, `J`, `K` e `L` vengono utilizzati per navigare, non per scrivere. In questa *modalità di navigazione*, puoi premere `I` per accedere alla *modalità di inserimento*. A quel punto, puoi scrivere normalmente. Per uscire dalla *modalità di inserimento*, puoi premere `Esc` che ti farà tornare alla *modalità di navigazione*. Dalla *modalità di navigazione*, puoi premere `:` per accedere alla *modalità di comando*. Da questa modalità è possibile salvare, eliminare, uscire o modificare le opzioni.

Sebbene `vi` abbia una curva di apprendimento, nel tempo le diverse modalità possono consentire a un utente esperto di diventare più efficiente di quanto sarebbe invece con altri editor.

nano

`nano` è uno strumento più recente, progettato per essere semplice e più facile da usare rispetto a `vi`. `nano` non prevede modalità differenti. Anzi: un utente all'avvio può iniziare a scrivere e può usare `Ctrl` per accedere agli strumenti mostrati nella parte inferiore dello schermo.

```
                        [ Welcome to nano.  For basic help, type Ctrl+G. ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line M-E Redo
```

Gli editor di testo sono solo una questione di preferenze personali e l'editor che scegli di utilizzare non farà alcuna differenza in questa lezione. Acquisire però familiarità e sentirsi a proprio agio con uno o più editor di testo ripagherà certamente in futuro.

Variabili

Le variabili sono una parte importante di qualsiasi linguaggio di programmazione e lo stesso vale anche per Bash. Quando avvii una nuova sessione da terminale, la shell imposta per te alcune variabili. La variabile `PATH` ne è un esempio. Queste sono chiamate *variabili ambientali*, poichè generalmente definiscono le caratteristiche del nostro ambiente di shell. È possibile modificare e aggiungere variabili ambientali, ma per ora ci concentreremo su come impostare le variabili all'interno dei nostri script.

Modifichiamo il nostro script in questo modo:

```
#!/bin/bash

# This is our first comment. It is also good practice to comment all scripts.

username=Carol

echo "Hello $username!"
```

In questo caso abbiamo creato una *variabile* chiamata `username` e le abbiamo assegnato il *valore* `Carol`. Nota che non ci sono spazi tra il nome della variabile, il segno di uguale e il valore assegnato.

Nella riga successiva, abbiamo usato il comando `echo` con la variabile, ma con un segno di dollaro (`$`) davanti al nome della variabile. Questo è importante, poiché indica alla shell che desideriamo trattare `username` come una variabile *e non* come una normale parola. Inserendo `$username` nel nostro comando, indichiamo che vogliamo eseguire una *sostituzione*, sostituendo il *nome* della variabile con il *valore* a essa assegnato.

Eseguendo il nuovo script, otteniamo questo output:

```
$ ./new_script.sh
Hello Carol!
```

- I nomi delle variabili devono contenere solo caratteri alfanumerici o underscore (trattini bassi) e fanno distinzione tra lettere maiuscole e minuscole. `Username` e `username` sono trattate come due differenti variabili.
- La sostituzione di variabile può anche seguire il formato `${username}`, con l'aggiunta di `{}`. Anche questa forma è accettata.
- Le variabili in Bash hanno un *tipo implicito* e sono considerate stringhe. Ciò significa che eseguire funzioni matematiche in Bash è più complicato di quanto lo sarebbe in altri linguaggi di programmazione come C/C++:

```
#!/bin/bash

# This is our first comment. It is also good practice to comment all scripts.

username=Carol
x=2
```

```
y=4
z=$x+$y
echo "Hello $username!"
echo "$x + $y"
echo "$z"
```

```
$ ./new_script.sh
Hello Carol!
2 + 4
2+4
```

Uso delle Virgolette con le Variabili

Apportiamo la seguente modifica al valore della nostra variabile `username`:

```
#!/bin/bash

# This is our first comment. It is also good practice to comment all scripts.

username=Carol Smith

echo "Hello $username!"
```

L'esecuzione di questo script ci darà un errore:

```
$ ./new_script.sh
./new_script.sh: line 5: Smith: command not found
Hello !
```

Ricorda che Bash è un interprete e, come tale, *interpreta* il nostro script riga per riga. In questo caso interpreta correttamente `username=Carol` che assegna alla variabile `username` il valore `Carol`, ma poi interpreta lo spazio come la fine di quell'assegnamento e `Smith` come il nome di un comando. Per poter includere lo spazio e il nome `Smith` nel valore della nostra variabile, utilizziamo le virgolette doppie (`"`) attorno al nome.

```
#!/bin/bash

# This is our first comment. It is also good practice to comment all scripts.
```

```
username="Carol Smith"

echo "Hello $username!"
```

```
$ ./new_script.sh
Hello Carol Smith!
```

Una cosa importante da notare in Bash è che le virgolette doppie e le virgolette singole (`'`) si comportano in modo molto diverso. Le virgolette doppie sono considerate *deboli*, poichè consentono all'interprete di eseguire una sostituzione all'interno di esse. Le virgolette singole sono considerate *forti*, poichè impediscono qualsiasi sostituzione. Considera il seguente esempio:

```
#!/bin/bash

# This is our first comment. It is also good practice to comment all scripts.

username="Carol Smith"

echo "Hello $username!"
echo 'Hello $username!'
```

```
$ ./new_script.sh
Hello Carol Smith!
Hello $username!
```

Nel secondo comando `echo` viene impedito all'interprete di sostituire `$username` con `Carol Smith`; quindi l'output viene scritto in modo letterale.

Argomenti

Hai già familiarità con l'utilizzo degli argomenti nelle principali utility di Linux. Per esempio, `rm testfile` contiene sia l'eseguibile `rm` che un argomento chiamato `testfile`. Gli argomenti possono essere passati a uno script durante l'esecuzione e modificarne il comportamento. Sono facili da implementare.

```
#!/bin/bash

# This is our first comment. It is also good practice to comment all scripts.
```

```
username=$1

echo "Hello $username!"
```

Invece di assegnare un valore alla variabile `username` direttamente all'interno dello script, le assegniamo il valore di una nuova variabile `$1`, che si riferisce al valore del *primo argomento*.

```
$ ./new_script.sh Carol
Hello Carol!
```

I primi nove argomenti vengono gestiti in questo modo. È possibile gestire anche più di nove argomenti, ma questo esula dallo scopo della lezione. Ecco un esempio che usa solo due argomenti:

```
#!/bin/bash

# This is our first comment. It is also good practice to comment all scripts.

username1=$1
username2=$2
echo "Hello $username1 and $username2!"
```

```
$ ./new_script.sh Carol Dave
Hello Carol and Dave!
```

C'è una importante considerazione da fare quando si usano gli argomenti: nell'esempio sopra riportato, ci sono due argomenti: `Carol` e `Dave`, assegnati rispettivamente a `$1` e `$2`. Se, per esempio, mancasse il secondo argomento, la shell non genererebbe un errore, ma il valore di `$2` sarebbe semplicemente *null*, o nessun valore.

```
$ ./new_script.sh Carol
Hello Carol and !
```

Nel nostro caso, potrebbe essere una buona idea introdurre un po' di logica all'interno dello script in modo che differenti *condizioni* influenzino l'*output* che vogliamo stampare. Inizieremo introducendo un'altra utile variabile e poi passeremo alla creazione di *istruzioni if*.

Restituire il Numero di Argomenti

Mentre alcune variabili come `$1` e `$2` contengono il valore degli argomenti posizionali, un'altra variabile, `$#`, contiene il *numero di argomenti*.

```
#!/bin/bash

# This is our first comment. It is also good practice to comment all scripts.

username=$1

echo "Hello $username!"
echo "Number of arguments: $#."
```

```
$ ./new_script.sh Carol Dave
Hello Carol!
Number of arguments: 2.
```

Logica Condizionale

Nella programmazione, l'uso della logica condizionale è un argomento molto vasto e non verrà trattato in modo approfondito in questa lezione. Ci concentreremo sulla *sintassi* delle istruzioni condizionali in Bash, sintassi che differisce dalla maggior parte degli altri linguaggi di programmazione.

Cominciamo esaminando ciò che speriamo di ottenere. Abbiamo un semplice script in grado di stampare un saluto a un singolo utente; se venisse indicato più di un utente, dovrebbe essere stampato un messaggio di errore.

- La *condizione* che stiamo testando riguarda il numero di utenti, che è contenuto nella variabile `$#`. Vogliamo sapere se il valore di `$#` è `1`.
- Se la condizione è *vera*, l'*azione* intrapresa sarà salutare l'utente.
- Se la condizione è *falsa*, verrà stampato un messaggio di errore.

Ora che la logica è chiara, concentriamoci sulla *sintassi* necessaria per implementarla.

```
#!/bin/bash

# A simple script to greet a single user.
```

```
if [ $# -eq 1 ]
then
  username=$1

  echo "Hello $username!"
else
  echo "Please enter only one argument."
fi
echo "Number of arguments: $#."
```

La logica condizionale è contenuta tra `if` e `fi`. La condizione da verificare si trova tra la parentesi quadre `[]` e l'azione da intraprendere se la condizione è vera è indicata dopo `then`. Nota gli spazi tra le parentesi quadre e la logica in esse contenuta. Tralasciare questi spazi causerà errori.

Questo script mostrerà come output o il nostro saluto *o* il messaggio di errore, ma stamperà sempre la riga `Number of arguments`.

```
$ ./new_script.sh
Please enter only one argument.
Number of arguments: 0.
$ ./new_script.sh Carol
Hello Carol!
Number of arguments: 1.
```

Nota l'istruzione `if`: abbiamo usato `-eq` per fare un *confronto numerico*. In questo caso, stiamo verificando che il valore di `$#` sia *uguale* a uno. Gli altri confronti che possiamo eseguire sono:

`-ne`
: Diverso da

`-gt`
: Maggiore di

`-ge`
: Maggiore o uguale a

`-lt`
: Minore di

`-le`

Minore o uguale a

Esercizi Guidati

1. L'utente digita quanto segue nella propria shell:

```
$ PATH=~/scripts
$ ls
Command 'ls' is available in '/bin/ls'
The command could not be located because '/bin' is not included in the PATH environment
variable.
ls: command not found
```

   - Cosa ha fatto l'utente?

   - Quale comando può essere usato per aggiungere la nuova directory `~/scripts` al valore corrente di `PATH`?

2. Considera il seguente script. Nota che viene usato `elif` per verificare una seconda condizione:

```
>  /!bin/bash

> fruit1 = Apples
> fruit2 = Oranges

  if [ $1 -lt $# ]
  then
    echo "This is like comparing $fruit1 and $fruit2!"
> elif [$1 -gt $2 ]
  then
>   echo '$fruit1 win!'
  else
>   echo "Fruit2 win!"
> done
```

   - Le righe contrassegnate da un `>` contengono degli errori. Correggili.

3. Quale sarà l'output in ciascuno dei seguenti casi?

```
$ ./guided1.sh 3 0
```

```
$ ./guided1.sh 2 4
```

```
$ ./guided1.sh 0 1
```

Esercizi Esplorativi

1. Scrivi un semplice script che controlli se vengono passati esattamente due argomenti. In tal caso, stampa gli argomenti in ordine inverso. Considera questo esempio (nota: il tuo codice potrebbe avere un aspetto diverso, ma dovrebbe restituire lo stesso output):

```
if [ $1 == $number ]
then
  echo "True!"
fi
```

2. Questo codice è corretto ma non rappresenta un confronto numerico. Facendo una ricerca su Internet, scopri in che modo questo codice differisca dall'utilizzo di `-eq`.

3. Esiste una variabile ambientale che stampa la directory corrente. Usa `env` per scoprire il nome di questa variabile.

4. In base a quanto appreso nelle domande 2 e 3, scrivi un breve script che accetti un argomento. Quando viene passato, controlla se quell'argomento corrisponde al nome della directory corrente. In tal caso, stampa `yes`; altrimenti, stampa `no`.

Sommario

In questa lezione hai imparato:

- Come creare ed eseguire semplici script;
- Come usare uno shebang per specificare un interprete;
- Come impostare e utilizzare le variabili all'interno degli script;
- Come gestire gli argomenti negli script;
- Come costruire istruzioni `if`;
- Come confrontare numeri utilizzando operatori numerici.

Comandi utilizzati negli esercizi:

`echo`
: Stampa una stringa sullo standard output.

`env`
: Stampa tutte le variabili ambientali sullo standard output.

`which`
: Stampa il percorso assoluto di un comando.

`chmod`
: Modifica i permessi di un file.

Variabili speciali utilizzate negli esercizi:

`$1, $2, … $9`
: Contengono gli argomenti posizionali passati allo script.

`$#`
: Contiene il numero di argomenti passati allo script.

`$PATH`
: Contiene le directory in cui si trovano gli eseguibili utilizzati dal sistema.

Operatori utilizzati negli esercizi:

`-ne`

Diverso da

`-gt`

Maggiore di

`-ge`

Maggiore o uguale a

`-lt`

Minore di

`-le`

Minore o uguale a

Risposte agli Esercizi Guidati

1. L'utente digita quanto segue nella propria shell:

```
$ PATH=~/scripts
$ ls
Command 'ls' is available in '/bin/ls'
The command could not be located because '/bin' is not included in the PATH environment
variable.
ls: command not found
```

   - Cosa ha fatto l'utente?

     L'utente ha sovrascritto il contenuto di PATH con la directory `~/scripts`. Il comando `ls` non può più essere trovato, poiché non è contenuto in PATH. Nota che questa modifica influisce solo sulla sessione corrente; disconnettiti e accedi nuovamente per annullare la modifica.

   - Quale comando può essere usato per aggiungere la nuova directory `~/scripts` al valore corrente di `PATH`?

     `PATH=$PATH:~/scripts`

2. Considera il seguente script. Nota che viene usato `elif` per verificare una seconda condizione:

```
>  /!bin/bash

> fruit1 = Apples
> fruit2 = Oranges

  if [ $1 -lt $# ]
  then
    echo "This is like comparing $fruit1 and $fruit2!"
> elif [$1 -gt $2 ]
  then
>   echo '$fruit1 win!'
  else
>   echo "Fruit2 win!"
> done
```

   - Le righe contrassegnate da un `>` contengono degli errori. Correggili.

```
#!/bin/bash

fruit1=Apples
fruit2=Oranges

if [ $1 -lt $# ]
then
  echo "This is like comparing $fruit1 and $fruit2!"
elif [ $1 -gt $2 ]
then
  echo "$fruit1 win!"
else
  echo "$fruit2 win!"
fi
```

3. Quale sarà l'output in ciascuno dei seguenti casi?

```
$ ./guided1.sh 3 0
```

`Apples win!`

```
$ ./guided1.sh 2 4
```

`Oranges win!`

```
$ ./guided1.sh 0 1
```

`This is like comparing Apples and Oranges!`

Risposte agli Esercizi Esplorativi

1. Scrivi un semplice script che controlli se vengono passati esattamente due argomenti. In tal caso, stampa gli argomenti in ordine inverso. Considera questo esempio (nota: il tuo codice potrebbe avere un aspetto diverso, ma dovrebbe restituire lo stesso output):

```
if [ $1 == $number ]
then
  echo "True!"
fi
```

```
#!/bin/bash

if [ $# -ne 2 ]
then
  echo "Error"
else
  echo "$2 $1"
fi
```

2. Questo codice è corretto ma non rappresenta un confronto numerico. Facendo una ricerca su Internet, scopri in che modo questo codice differisca dall'utilizzo di `-eq`.

   Con `==` si confrontano *stringhe*, ovvero: se i caratteri di entrambe le variabili corrispondono esattamente, la condizione è vera.

`abc == abc`	*vero*
`abc == ABC`	*falso*
`1 == 1`	*vero*
`1+1 == 2`	*falso*

   I confronti tra stringhe causano un comportamento anomalo se usati in condizioni di test con numeri.

3. Esiste una variabile ambientale che stampa la directory corrente. Usa `env` per scoprire il nome di questa variabile.

   `PWD`

4. In base a quanto appreso nelle domande 2 e 3, scrivi un breve script che accetti un argomento. Quando viene passato, controlla se quell'argomento corrisponde al nome della directory corrente. In tal caso, stampa `yes`; altrimenti, stampa `no`.

```
#!/bin/bash

if [ "$1" == "$PWD" ]
then
  echo "yes"
else
  echo "no"
fi
```

3.3 Lezione 2

Certificazione:	Linux Essentials
Versione:	1.6
Argomento:	3 Il Potere della Command Line
Obiettivo:	3.3 Trasformare i Comandi in uno Script
Lezione:	2 di 2

Introduzione

Nell'ultima lezione abbiamo utilizzato questo semplice esempio per mostrare uno script Bash:

```
#!/bin/bash

# A simple script to greet a single user.

if [ $# -eq 1 ]
then
  username=$1

  echo "Hello $username!"
else
  echo "Please enter only one argument."
fi
echo "Number of arguments: $#."
```

- Tutti gli script dovrebbero iniziare con uno *shebang* che definisce il percorso dell'interprete.

- Tutti gli script dovrebbero includere dei commenti per descrivere il loro utilizzo.
- Questo specifico script funziona con un *argomento* che gli viene passato al momento della sua invocazione.
- Questo script contiene una *istruzione if* che testa le condizioni della variabile interna (built-in) `$#` che contiene il numero di argomenti.
- Se il numero di argomenti passati allo script è uguale a 1, allora il valore del primo argomento viene assegnato a una nuova variabile chiamata `username` e lo script stampa un saluto all'utente. In caso contrario, viene visualizzato un messaggio di errore.
- Infine, lo script stampa il numero di argomenti. Questo è utile per il debug.

Questo è un esempio utile per iniziare a spiegare alcune delle altre caratteristiche degli script Bash.

Codici di Uscita

Avrai notato che il nostro script ha due possibili stati: o stampa `"Hello <user>!"` o stampa un messaggio di errore. Questo è piuttosto normale per molte delle nostre principali utility. Considera `cat`, con cui senza dubbio stai diventando molto familiare.

Confrontiamo ora una situazione in cui `cat` viene eseguito con successo con una in cui dà errore. Ricorda che il nostro esempio sopra riportato è uno script chiamato `new_script.sh`.

```
$ cat -n new_script.sh

     1  #!/bin/bash
     2
     3  # A simple script to greet a single user.
     4
     5  if [ $# -eq 1 ]
     6  then
     7    username=$1
     8
     9    echo "Hello $username!"
    10  else
    11    echo "Please enter only one argument."
    12  fi
    13  echo "Number of arguments: $#."
```

Questo comando viene eseguito con successo; nota che l'opzione `-n` ha anche stampato i numeri di riga. Questi sono molto utili durante il debug degli script, ma tieni presente che *non* fanno parte

dello script.

Controlliamo ora il valore di una nuova variabile interna (built-in): `$?`. Per il momento, nota solamente il valore di output:

```
$ echo $?
0
```

Consideriamo ora una situazione in cui `cat` non viene eseguito con successo. Per prima cosa osserviamo il messaggio di errore e poi controlliamo il valore di `$?`.

```
$ cat -n dummyfile.sh
cat: dummyfile.sh: No such file or directory
$ echo $?
1
```

La spiegazione per questo comportamento è la seguente: qualsiasi esecuzione dell'utility `cat` restituisce un *codice di uscita*. Un codice di uscita indica se il comando è stato eseguito con successo o se si è verificato un errore. Un codice di uscita pari a *zero* indica che il comando è stato completato correttamente. Questo è vero per quasi tutti i comandi Linux con cui hai a che fare. Qualsiasi altro codice di uscita indica un errore di qualche tipo. Il codice di uscita dell'*ultimo comando eseguito* è memorizzato nella variabile `$?`.

I codici di uscita, di solito, non sono controllati dagli utenti in carne e ossa, ma sono molto utili durante la redazione di script. Considera uno script che copia i file su un'unità di rete remota: esistono molte cause per le quali l'attività di copia potrebbe non andare a buon fine: per esempio, la nostra macchina locale potrebbe non essere collegata alla rete o l'unità remota potrebbe essere piena. Controllando il codice di uscita della nostra utility di copia possiamo avvisare l'utente di eventuali problemi durante l'esecuzione dello script.

È buona norma implementare i codici di uscita ed è ciò che faremo ora. Abbiamo due percorsi nel nostro script: un successo e un errore. Usiamo zero per indicare il successo e uno per indicare l'errore.

```
1  #!/bin/bash
2
3  # A simple script to greet a single user.
4
5  if [ $# -eq 1 ]
6  then
```

```
 7    username=$1
 8
 9    echo "Hello $username!"
10    exit 0
11  else
12    echo "Please enter only one argument."
13    exit 1
14  fi
15  echo "Number of arguments: $#."
```

```
$ ./new_script.sh Carol
Hello Carol!
$ echo $?
0
```

Nota che il comando `echo` alla riga 15 viene completamente ignorato. Infatti, `exit` termina immediatamente lo script; quindi questa riga non viene mai raggiunta.

Gestire più Argomenti

Per ora il nostro script può gestire un solo nome utente alla volta. Qualsiasi numero di argomenti diverso da 1 causerà un errore. Esaminiamo come poter rendere questo script più versatile.

Il primo istinto di un utente potrebbe essere quello di utilizzare più variabili posizionali come, per esempio, `$2`, `$3` e così via. Sfortunatamente, non possiamo prevedere il numero di argomenti che un utente potrebbe voler utilizzare. Per risolvere questo problema può essere utile introdurre altre variabili interne (built-in).

Modifichiamo la logica del nostro script: in caso ci siano 0 argomenti, dovrebbe essere generato un errore, ma un qualsiasi altro numero di argomenti renderebbe lo script corretto. Questo nuovo script si chiamerà `friendly2.sh`.

```
 1  #!/bin/bash
 2
 3  # a friendly script to greet users
 4
 5  if [ $# -eq 0 ]
 6  then
 7    echo "Please enter at least one user to greet."
 8    exit 1
 9  else
```

```
    10    echo "Hello $@!"
    11    exit 0
    12  fi
```

```
$ ./friendly2.sh Carol Dave Henry
Hello Carol Dave Henry!
```

Esistono due variabili interne (built-in) che contengono tutti gli argomenti passati allo script: `$@` e `$*`. Nella maggior parte dei casi entrambe si comportano allo stesso modo. Bash *analizza* gli argomenti e li separa quando incontra uno spazio. In pratica, il contenuto di `$@` ha questo aspetto:

`0`	`1`	`2`
`Carol`	`Dave`	`Henry`

Se hai familiarità con altri linguaggi di programmazione potresti conoscere questo tipo di variabile come un *array*. Gli array in Bash possono essere creati semplicemente inserendo uno spazio tra gli elementi, come per la variabile `FILES` nello script `arraytest` qui sotto:

```
FILES="/usr/sbin/accept /usr/sbin/pwck/ usr/sbin/chroot"
```

La variabile contiene un elenco di diversi elementi. Per ora non è molto utile, poichè non abbiamo ancora introdotto un modo per gestire singolarmente questi elementi.

Cicli For

Facciamo riferimento all'esempio `arraytest` mostrato in precedenza. Se ricordi, in questo esempio abbiamo specificato un nostro array chiamato `FILES`. Ciò di cui abbiamo bisogno è un modo per "spacchettare" questa variabile e accedere a ogni singolo valore, uno dopo l'altro. Per fare questo, useremo una struttura chiamata *ciclo for*, che è presente in tutti i linguaggi di programmazione. Faremo riferimento a due variabili: una è l'intervallo e l'altra è il valore individuale su cui stiamo attualmente lavorando. Questo è lo script nella sua interezza:

```
#!/bin/bash

FILES="/usr/sbin/accept /usr/sbin/pwck/ usr/sbin/chroot"

for file in $FILES
do
  ls -lh $file
```

```
done
```

```
$ ./arraytest
lrwxrwxrwx 1 root root 10 Apr 24 11:02 /usr/sbin/accept -> cupsaccept
-rwxr-xr-x 1 root root 54K Mar 22 14:32 /usr/sbin/pwck
-rwxr-xr-x 1 root root 43K Jan 14 07:17 /usr/sbin/chroot
```

Se fai nuovamente riferimento all'esempio `friendly2.sh` sopra riportato noterai che stiamo lavorando con una serie di valori contenuti all'interno di un'unica variabile `$@`. Per maggiore chiarezza chiameremo quest'ultima variabile `username`. Il nostro script ora ha questo aspetto:

```
 1  #!/bin/bash
 2
 3  # a friendly script to greet users
 4
 5  if [ $# -eq 0 ]
 6  then
 7    echo "Please enter at least one user to greet."
 8    exit 1
 9  else
10    for username in $@
11    do
12      echo "Hello $username!"
13    done
14    exit 0
15  fi
```

Ricorda che la variabile che definisci può avere un qualunque nome e che tutte le righe all'interno di `do… done` verranno eseguite una volta per ogni elemento dell'array. Osserviamo l'output del nostro script:

```
$ ./friendly2.sh Carol Dave Henry
Hello Carol!
Hello Dave!
Hello Henry!
```

Supponiamo ora di voler rendere il nostro output un po' più umano, facendo in modo che il saluto appaia su un'unica riga.

```
 1  #!/bin/bash
```

```
 2
 3  # a friendly script to greet users
 4
 5  if [ $# -eq 0 ]
 6  then
 7    echo "Please enter at least one user to greet."
 8    exit 1
 9  else
10    echo -n "Hello $1"
11    shift
12    for username in $@
13    do
14      echo -n ", and $username"
15    done
16    echo "!"
17    exit 0
18  fi
```

Un paio di note:

- L'uso di `-n` con `echo` *elimina il carattere di nuova riga* dopo la stampa. Questo significa che tutto l'output verrà stampato sulla stessa riga e il carattere di nuova riga verrà stampato solo dopo il `!` della riga 16.
- Il comando `shift` rimuove il primo elemento del nostro array, in modo tale che questo:

`0`	`1`	`2`
`Carol`	`Dave`	`Henry`

diventi questo:

`0`	`1`
`Dave`	`Henry`

Diamo un'occhiata all'output:

```
$ ./friendly2.sh Carol
Hello Carol!
$ ./friendly2.sh Carol Dave Henry
Hello Carol, and Dave, and Henry!
```

Usare le Espressioni Regolari per Eseguire il Controllo degli Errori

Potremmo voler verificare tutti gli argomenti che inserisce l'utente. Per esempio, potremmo verificare che tutti i nomi passati a `friendly2.sh` contengano *solo lettere*, in modo tale che qualsiasi carattere speciale o numero causi un errore. Per eseguire questo controllo degli errori useremo `grep`.

Ricorda che possiamo usare le espressioni regolari con `grep`.

```
$ echo Animal | grep "^[A-Za-z]*$"
Animal
$ echo $?
0
```

```
$ echo 4n1ml | grep "^[A-Za-z]*$"
$ echo $?
1
```

`^` e `$` indicano rispettivamente l'inizio e la fine della riga. `[A-Za-z]` indica un intervallo di lettere, maiuscole o minuscole. `*` è un *quantificatore* che modifica il nostro intervallo di lettere in modo che si abbia corrispondenza per 0 o un qualsiasi numero di lettere. In sintesi: il nostro `grep` verrà eseguito con successo se l'input è costituito *solo* da lettere, e darà errore in caso contrario.

La prossima cosa da notare è che `grep` restituisce i codici di uscita in base al fatto che ci sia o meno corrispondenza con quanto inserito. Se c'è corrispondenza restituisce `0`, mentre se non c'è corrispondenza restituisce `1`. Possiamo quindi utilizzarlo per testare gli argomenti all'interno del nostro script.

```
 1  #!/bin/bash
 2
 3  # a friendly script to greet users
 4
 5  if [ $# -eq 0 ]
 6  then
 7    echo "Please enter at least one user to greet."
 8    exit 1
 9  else
10    for username in $@
11    do
```

```
12      echo $username | grep "^[A-Za-z]*$" > /dev/null
13      if [ $? -eq 1 ]
14      then
15        echo "ERROR: Names must only contains letters."
16        exit 2
17      else
18        echo "Hello $username!"
19      fi
20    done
21    exit 0
22  fi
```

Nella riga 12 redirigiamo lo standard output verso `/dev/null`, il che rappresenta un semplice modo per sopprimere tale output. Non vogliamo vedere alcun output del comando `grep`: vogliamo solo testare il suo codice di uscita, cosa che facciamo nella riga 13. Nota inoltre che stiamo usando un codice di uscita pari a `2` per indicare un argomento non valido. In genere è buona norma utilizzare codici di uscita diversi per indicare errori diversi: in questo modo, un utente esperto può utilizzare questi codici di uscita per individuare e risolvere i problemi.

```
$ ./friendly2.sh Carol Dave Henry
Hello Carol!
Hello Dave!
Hello Henry!
$ ./friendly2.sh 42 Carol Dave Henry
ERROR: Names must only contains letters.
$ echo $?
2
```

Esercizi Guidati

1. Osserva il contenuto di `script1.sh` qui sotto riportato:

```
#!/bin/bash

if [ $# -lt 1 ]
then
  echo "This script requires at least 1 argument."
  exit 1
fi

echo $1 | grep "^[A-Z]*$" > /dev/null
if [ $? -ne 0 ]
then
  echo "no cake for you!"
  exit 2
fi

echo "here's your cake!"
exit 0
```

   Quale sarà l'output dei seguenti comandi?

   - `./script1.sh`
   - `echo $?`
   - `./script1.sh cake`
   - `echo $?`
   - `./script1.sh CAKE`
   - `echo $?`

2. Osserva il contenuto del file `script2.sh`:

```
for filename in $1/*.txt
do
   cp $filename $filename.bak
done
```

Descrivi lo scopo di questo script.

Esercizi Esplorativi

1. Crea uno script che accetti un numero qualsiasi di argomenti dall'utente e stampi solamente gli argomenti che hanno come valore un numero maggiore di 10.

Sommario

In questa lezione hai imparato:

- Cosa sono i codici di uscita, cosa significano e come si implementano;
- Come controllare il codice di uscita di un comando;
- Cosa sono i cicli `for` e come usarli con gli array;
- Come usare `grep`, le espressioni regolari e i codici di uscita per controllare l'input dell'utente negli script.

Comandi utilizzati negli esercizi:

`shift`
: Rimuove il primo elemento di un array.

Variabili speciali:

`$?`
: Contiene il codice di uscita dell'ultimo comando eseguito.

`$@`, **`$*`**
: Contiene tutti gli argomenti passati allo script, sotto forma di un array.

Risposte agli Esercizi Guidati

1. Osserva il contenuto di `script1.sh` qui sotto riportato:

```
#!/bin/bash

if [ $# -lt 1 ]
then
  echo "This script requires at least 1 argument."
  exit 1
fi

echo $1 | grep "^[A-Z]*$" > /dev/null
if [ $? -ne 0 ]
then
  echo "no cake for you!"
  exit 2
fi

echo "here's your cake!"
exit 0
```

   Quale sarà l'output dei seguenti comandi?

   - Comando: `./script1.sh`

     Output: `This script requires at least 1 argument.`

   - Comando: `echo $?`

     Output: `1`

   - Comando: `./script1.sh cake`

     Output: `no cake for you!`

   - Comando: `echo $?`

     Output: `2`

   - Comando: `./script1.sh CAKE`

     Output: `here's your cake!`

- Comando: `echo $?`

  Output: `0`

2. Osserva il contenuto del file `script2.sh`:

```
for filename in $1/*.txt
do
   cp $filename $filename.bak
done
```

Descrivi lo scopo di questo script.

Questo script crea delle copie di backup di tutti i file che terminano con `.txt` che si trovano in una sottodirectory definita nel primo argomento.

Risposte agli Esercizi Esplorativi

1. Crea uno script che accetti un numero qualsiasi di argomenti dall'utente e stampi solamente gli argomenti che hanno come valore un numero maggiore di 10.

```
#!/bin/bash

for i in $@
do
  echo $i | grep "^[0-9]*$" > /dev/null
  if [ $? -eq 0 ]
  then
    if [ $i -gt 10 ]
    then
      echo -n "$i "
    fi
  fi
done
echo ""
```

Linux Professional Institute

Argomento 4: Il Sistema Operativo Linux

 | Questa opera è concessa in licenza CC BY-NC-ND 4.0. |

4.1 Scelta di un Sistema Operativo

Obiettivi LPI di riferimento

Linux Essentials version 1.6, Exam 010, Objective 4.1

Peso

1

Aree di Conoscenza Chiave

- Differenze tra Windows, OS X e Linux
- Gestione del ciclo di vita di una distribuzione

Di seguito è riportato un elenco parziale dei file, dei termini e dei comandi utilizzati

- GUI contro command line, configurazione desktop
- Cicli di manutenzione, versioni beta e stabili

4.1 Lezione 1

Certificazione:	Linux Essentials
Versione:	1.6
Argomento:	4 Il Sistema Operativo Linux
Obiettivo:	4.1 Scelta di un Sistema Operativo
Lezione:	1 di 1

Introduzione

Sia che utilizzi il tuo computer a casa, all'università o all'interno di un'azienda, devi comunque decidere il suo sistema operativo. Puoi prendere tu questa decisione, specialmente se si tratta del tuo computer, ma potresti anche essere responsabile della scelta dei sistemi per la tua azienda. Come sempre, essere ben informati sulle opzioni disponibili ti aiuterà a prendere una decisione responsabile. In questa lezione ti mostreremo alcune alternative da tenere in considerazione nella scelta del sistema operativo.

Cos'è un Sistema Operativo

Una delle prime cose che dobbiamo assicurarci, prima di iniziare il nostro viaggio nella scelta di un sistema operativo, è capire cosa intendiamo con questo termine. Il sistema operativo è il cuore del tuo computer ed è ciò che rende possibile eseguire le applicazioni al suo interno o su di esso. Inoltre, il sistema operativo contiene i driver per accedere all'hardware del computer come, per esempio, dischi e partizioni, schermi, tastiere, schede di rete e così via. Di solito abbreviamo il termine sistema operativo semplicemente con *SO* (in inglese OS, che è l'acronimo di *Operating System*). Oggigiorno esistono molti sistemi operativi disponibili sia per uso aziendale sia per uso domestico. Per facilitare la scelta, possiamo raggrupparli come segue:

- Sistemi Operativi basati su Linux
  - Linux per le aziende (Enterprise Linux)
  - Linux per uso privato (Consumer Linux)
- UNIX
- macOS
- Sistemi Operativi basati su Windows
  - Sistemi Windows per server (Windows Server)
  - Sistemi Windows per desktop (Windows Desktop)

Scegliere una Distribuzione Linux

Il Kernel Linux e le Distribuzioni Linux

Quando si parla di distribuzioni Linux, il sistema operativo è Linux. Linux è il *kernel*, il cuore di ogni distribuzione. Il software del kernel Linux è mantenuto da un gruppo di individui, guidati da Linus Torvalds, che lavorano sul kernel Linux per un consorzio industriale chiamato The Linux Foundation.

NOTE Il kernel Linux è stato sviluppato per la prima volta da Linus Torvalds, uno studente finlandese, nel 1991. La prima versione rilasciata del kernel sotto la GNU General Public License versione 2 (GPLv2) è stata la 0.12 nel 1992.

Kernel Linux

Come accennato in precedenza, tutte le distribuzioni Linux eseguono lo stesso sistema operativo: Linux.

Distribuzione Linux

Quando le persone parlano di Red Hat Linux o Ubuntu Linux, si riferiscono a una *distribuzione Linux* specifica. Una distribuzione Linux è provvista di un kernel Linux e di un ambiente che rende il kernel utilizzabile in modo che si possa interagire con esso. Come minimo avremo bisogno di una shell a Command Line, come per esempio Bash, e di un insieme di comandi di base che ci consentano di accedere e gestire il sistema. Nella maggior parte dei casi, ovviamente, una distribuzione Linux includerà un ambiente desktop completo, come Gnome o KDE.

Anche se ogni distribuzione Linux esegue il sistema operativo Linux, le distribuzioni possono variare e variano in base alla versione del sistema operativo utilizzato. Con questa affermazione intendiamo *la versione del kernel Linux che è in uso* all'avvio della distribuzione.

TIP

Se hai accesso alla Command Line Linux, puoi facilmente controllare la versione del kernel Linux che stai utilizzando semplicemente visualizzando la *kernel release*:

```
$ uname -r
4.15.0-1019-aws
```

Tipi di Distribuzioni Linux

Può sembrare una scelta ovvia utilizzare sempre l'ultima versione del kernel Linux, ma non è così semplice. Approssimativamente, possiamo suddividere le distribuzioni Linux in tre categorie:

- Distribuzioni Linux aziendali (di livello Enterprise)
  - Red Hat Enterprise Linux
  - CentOS
  - SUSE Linux Enterprise Server
  - Debian GNU/Linux
  - Ubuntu LTS
- Distribuzioni Linux per uso privato (di livello Consumer)
  - Fedora
  - Ubuntu non-LTS
  - openSUSE
- Distribuzioni Linux sperimentali e per hacker
  - Arch
  - Gentoo

Questo, ovviamente, è solo un piccolo sottoinsieme delle possibili distribuzioni. Ciò che è importante è la differenza tra le distribuzioni *aziendali* (*di livello Enterprise*), *per uso privato* (*di livello Consumer*) e *sperimentali*, e il motivo della loro esistenza.

Distribuzioni Linux Aziendali (di Livello Enterprise)

Distribuzioni come, per esempio, CentOS (*Community Enterprise OS*) sono progettate per essere implementate all'interno di grandi organizzazioni che utilizzano hardware a livello enterprise. Le esigenze delle grandi aziende sono molto diverse da quelle delle piccole aziende e da quelle di coloro che usano Linux per hobby o in ambito domestico. Per garantire la disponibilità dei propri servizi le aziende necessitano di requisiti più elevati in termini di stabilità a livello

hardware e software. Pertanto le distribuzioni Linux aziendali di livello enterprise tendono a includere versioni meno recenti del kernel e degli altri software che funzionano in modo più affidabile. In genere le distribuzioni apportano aggiornamenti importanti a queste versioni stabili, come per esempio correzioni di sicurezza. In compenso, le distribuzioni Linux aziendali di livello enterprise potrebbero non supportare l'hardware *consumer* più recente e potrebbero includere versioni meno recenti dei pacchetti software. Tuttavia, come le distribuzioni Linux di livello consumer, anche quelle di livello enterprise tendono a scegliere componenti hardware "maturi" e a creare i propri servizi su versioni stabili del software.

Distribuzioni Linux per Uso Privato (di Livello Consumer)

Distribuzioni come Ubuntu sono più indicate per piccole imprese o per coloro che usano Linux in ambito domestico o per hobby. Pertanto, è probabile che utilizzino anche l'hardware più recente disponibile per i sistemi di livello consumer. Questi sistemi avranno bisogno dei driver più recenti per ottenere il massimo dal nuovo hardware, ma è improbabile che la maturità sia dell'hardware che dei driver soddisfi le esigenze delle grandi imprese. Per il mercato consumer, tuttavia, l'ultimo kernel disponibile è esattamente ciò che serve, assieme ai driver più recenti anche se non sono stati adeguatamente testati. I kernel Linux più recenti includeranno gli ultimi driver per supportare l'hardware più recente che potrebbe essere usato. Data la crescita di Linux osservata nel mercato dei giochi, è estremamente importante che i driver più recenti siano disponibili per questi utenti.

NOTE

Alcune distribuzioni come Ubuntu prevedono sia versioni di livello consumer che contengono software recente e ricevono aggiornamenti per un periodo di tempo piuttosto breve, sia le così dette versioni di supporto a lungo termine (in breve LTS) che sono più adatte in ambito aziendale.

Distribuzioni Linux Sperimentali e per Hacker

Distribuzioni come, per esempio, Arch Linux o Gentoo Linux sono all'avanguardia della tecnologia. Contengono le versioni più recenti del software, anche se hanno ancora bug e funzionalità non testate. In compenso, queste distribuzioni tendono a utilizzare un modello di rilascio progressivo che consente loro di distribuire aggiornamenti in qualsiasi momento. Queste distribuzioni sono utilizzate da utenti avanzati che desiderano avere sempre il software più recente, che sono consapevoli che il funzionamento può interrompersi in qualsiasi momento e che, quando questo accade, sono in grado di riparare i loro sistemi.

In sintesi: quando si prende in considerazione Linux come sistema operativo, se si utilizza hardware di livello enterprise in ambiente server o desktop, è possibile scegliere distribuzioni Linux di livello enterprise o consumer. Se si utilizza hardware di livello consumer e si vuole sfruttare al massimo le ultime innovazioni hardware, è probabile che sia necessaria una distribuzione Linux altrettanto recente per soddisfare le esigenze dell'hardware.

Alcune distribuzioni Linux sono correlate tra loro. Ubuntu, per esempio, è basato su Debian Linux e utilizza lo stesso sistema di pacchetti, DPKG. Fedora, per fare un altro esempio, è una sorta di piattaforma di test per RedHat Enterprise Linux, in cui possono essere esplorate e verificate le potenziali funzionalità delle future versioni di RHEL prima che vengano rese disponibili nella distribuzione enterprise.

Oltre alle distribuzioni Linux che abbiamo menzionato ce ne sono molte altre. Un vantaggio derivante dal fatto che Linux sia un software open source è che molte persone possono svilupparlo come meglio credono: per questo abbiamo molte centinaia di distribuzioni. Per una panoramica, puoi visitare The Distro Watch Web Site. I manutentori del sito web elencano le prime 100 distribuzioni Linux più scaricate, permettendoti di confrontare e vedere ciò che è attualmente popolare.

Ciclo di Vita del Supporto Linux

Come ci si potrebbe aspettare, le distribuzioni Linux di livello enterprise hanno un supporto più lungo, in termini di durata, rispetto alle versioni consumer o community di Linux. Per esempio Red Hat Enterprise Linux ha un supporto per 10 anni. Red Hat Enterprise Linux 8 è stato rilasciato a maggio 2019, mentre gli aggiornamenti software e il supporto saranno disponibili fino a maggio 2029.

Le versioni consumer, in genere, hanno il supporto della comunità solo attraverso i forum. Gli aggiornamenti software sono generalmente disponibili per tre rilasci. Se prendiamo Ubuntu come esempio, al momento della stesura di questa lezione, la versione più recente disponibile è la 19.04 con aggiornamenti tramite il rilascio della versione 19.10 e con termine a gennaio 2020. Ubuntu offre anche versioni con supporto a lungo termine, note come versioni *LTS*, che hanno 5 anni di supporto dal primo rilascio. La versione LTS corrente è la 18.04 che avrà aggiornamenti software fino al 2023. Queste versioni LTS rendono Ubuntu una possibile alternativa in ambito aziendale grazie al supporto commerciale fornito da Canonical (la società dietro il marchio Ubuntu) o da società di consulenza indipendenti.

NOTE

Le distribuzioni Ubuntu utilizzano i numeri di versione basati sulla data secondo il formato YY.MM (YY rappresenta le ultime 2 cifre dell'anno, mentre MM rappresenta il mese): per esempio, la versione 19.04 è stata rilasciata nell'aprile 2019.

Linux come Sistema Desktop

L'utilizzo di Linux come sistema desktop può essere più difficile in un'azienda in cui il supporto desktop è incentrato su sistemi operativi commerciali. Tuttavia non è solo il supporto che potrebbe rivelarsi difficoltoso. Un cliente aziendale potrebbe anche aver fatto grandi investimenti

in soluzioni software legate a specifici sistemi operativi desktop. Detto questo, ci sono molti esempi di integrazione di desktop Linux in grandi organizzazioni e aziende, come Amazon che ha anche una propria distribuzione: Amazon Linux 2. Questa viene utilizzata sulla piattaforma cloud AWS, ma anche internamente, sia per i server sia per i desktop.

Usare Linux in una piccola azienda o in un contesto domestico diventa molto più semplice e può essere un'esperienza gratificante, eliminando la necessità di licenze e consentendoti di scoprire la grande quantità di software gratuito e open source disponibile per Linux. Scoprirai anche che sono disponibili molti ambienti desktop diversi. I più comuni sono Gnome e KDE, ma ne esistono molti altri. La decisione dipende solamente dai tuoi gusti personali.

Utilizzo di Linux sui Server

L'utilizzo di Linux come sistema operativo per i server è molto comune nel settore aziendale. I server sono gestiti da ingegneri specializzati in Linux. Quindi, anche se ci sono migliaia di utenti, questi non hanno bisogno di saper nulla sui server a cui si connettono. Il sistema operativo del server non è importante per loro e, in generale, le applicazioni client non avranno differenze tra Linux e gli altri sistemi operativi lato *back-end*. È anche vero che, man mano che più applicazioni vengono virtualizzate o inserite in container all'interno di cloud locali e remoti, più il sistema operativo viene "mascherato" ed è probabile che il sistema operativo integrato sia Linux.

Linux nel Cloud

Un'altra opportunità per acquisire familiarità con Linux è implementarlo all'interno di uno dei tanti cloud pubblici disponibili. La creazione di un account con uno dei tanti provider cloud ti consentirà di utilizzare molte distribuzioni Linux differenti tra loro in modo rapido e semplice.

Sistemi Operativi non Linux

Per quanto incredibile possa sembrare, ci sono sistemi operativi che non sono basati sul kernel Linux. Certo, nel corso degli anni ce ne sono stati molti e alcuni sono caduti nel dimenticatoio, ma hai ancora a disposizione delle alternative, per uso domestico o per l'ufficio.

Unix

Prima che ci fosse Linux, come sistema operativo c'era Unix. Unix veniva venduto insieme all'hardware e ancora oggi sono disponibili sul mercato diversi sistemi Unix commerciali come AIX e HP-UX. Mentre Linux è stato fortemente ispirato da Unix (e dalla sua non disponibilità per determinati hardware), la famiglia di sistemi operativi BSD è basata direttamente su di esso. Oggi FreeBSD, NetBSD e OpenBSD, insieme ad altri sistemi BSD a essi correlati, sono disponibili come software libero.

Unix è stato ampiamente utilizzato in ambito aziendale, ma, con la crescita di Linux, si è assistito a un suo declino. Man mano che Linux è cresciuto e le offerte di supporto aziendale sono aumentate, Unix ha iniziato lentamente a scomparire. Solaris, in origine di Sun prima di passare a Oracle, è recentemente scomparso. Era uno dei più grandi sistemi operativi Unix utilizzati dalle società di telecomunicazioni, proclamato come *Unix per le Telecomunicazioni* (*Telco Grade Unix*).

I sistemi operativi Unix includono:

- AIX
- FreeBSD, NetBSD, OpenBSD
- HP-UX
- Irix
- Solaris

macOS

macOS (in precedenza OS X) di Apple risale al 2001. Basato in gran parte su BSD Unix e facendo uso della shell Bash, è un sistema intuitivo se sei abituato a usare sistemi operativi Unix o Linux. Se usi macOS puoi aprire l'applicazione terminale per accedere alla Command Line. Ancora una volta, eseguendo il comando `uname` è possibile controllare il sistema operativo rilevato:

```
$ uname -s
Darwin
```

NOTE In questo caso abbiamo utilizzato l'opzione `-s` per ottenere il nome del SO. In precedenza avevamo utilizzato `-r` per ottenere il numero di versione del kernel.

Microsoft Windows

Possiamo tranquillamente dire che la maggior parte dei desktop e dei laptop in circolazione è basata su Windows. Tale sistema operativo è stato davvero un successo e ha dominato per anni il mercato desktop. Sebbene sia un software proprietario e non sia gratuito, spesso la licenza del sistema operativo è inclusa nell'acquisto dell'hardware, facilitando la scelta dell'utente. Ovviamente i fornitori di hardware e software supportano ampiamente Windows e molte applicazioni open source sono disponibili anche per questo sistema operativo. Il futuro di Windows non sembra così luminoso come il passato, però con la diminuzione delle vendite di desktop e laptop, l'attenzione si è spostata sul mercato dei tablet e dei telefoni cellulari. Questo mercato è dominato da Apple e Android ed è difficile per Microsoft guadagnare terreno.

Come piattaforma server, Microsoft ora consente ai suoi clienti di scegliere tra una GUI (*Graphical User Interface*) e una versione solo a Command Line. La separazione della GUI e della Command Line è importante. La maggior parte delle volte viene caricata la GUI dei Microsoft Servers meno recenti, ma nessuno la utilizza. Considera un controller di dominio di Active Directory... gli utenti lo usano sempre per autenticarsi nel dominio, ma è gestito in remoto dai desktop degli amministratori e non dal server.

Esercizi Guidati

1. Quale programma rappresenta il componente comune a tutte le distribuzioni Linux?

CentOS	
Red Hat	
Ubuntu	
Kernel Linux	
CoreOS	

2. Quale sistema operativo utilizza macOS di Apple?

OS X	
OSX	
Darwin	
MacOS	

3. Qual è la differenza tra una distribuzione Linux e il kernel Linux?

Il kernel fa parte di una distribuzione, mentre la distribuzione è intesa come la serie di applicazioni che circondano il kernel per renderlo utilizzabile	
Il kernel è la distribuzione Linux	
Tutte le distribuzioni che utilizzano lo stesso kernel sono uguali	

4. Quale dei seguenti è un ambiente desktop in Linux?

Mint	
Elementary	
Zorin	
Gnome	

5. Quale componente di un sistema operativo consente l'accesso all'hardware?

Driver	
Shell	
Servizio	
Applicazione	

Esercizi Esplorativi

1. Se hai accesso alla Command Line, trova il rilascio corrente del kernel del tuo sistema.

2. Utilizzando il tuo motore di ricerca preferito, individua i provider di cloud pubblico a tua disposizione. Questi potrebbero includere AWS, Google Cloud, Rackspace e molti altri. Scegline uno e controlla quali sistemi operativi sono disponibili per l'installazione.

Sommario

In questa lezione hai imparato a distinguere i diversi sistemi operativi più comuni. Abbiamo parlato di:

- Sistemi Operativi basati su Linux;
- UNIX;
- macOS;
- Sistemi Operativi basati su Windows.

All'interno della categoria Linux possiamo suddividere ulteriormente la selezione in: distribuzioni con supporto a lungo termine e distribuzioni con un ciclo di supporto più breve. Le versioni LTS sono più adatte per uso aziendale, mentre quelle con supporto a breve termine sono rivolte a utenti domestici e a coloro che usano Linux per hobby.

- Distribuzioni Linux aziendali (di livello Enterprise)
  - Red Hat Enterprise Linux
  - CentOS
  - SUSE Linux Enterprise Server
  - Debian GNU/Linux
  - Ubuntu LTS
- Distribuzioni Linux per uso privato (di livello Consumer)
  - Fedora
  - Ubuntu non-LTS
  - openSUSE
- Distribuzioni Linux sperimentali e per hacker
  - Arch
  - Gentoo

Risposte agli Esercizi Guidati

1. Quale programma rappresenta il componente comune a tutte le distribuzioni Linux?

CentOS	
Red Hat	
Ubuntu	
Kernel Linux	X
CoreOS	

2. Quale sistema operativo utilizza macOS di Apple?

OS X	
OSX	
Darwin	X
MacOS	

3. Qual è la differenza tra una distribuzione Linux e il kernel Linux?

Il kernel fa parte di una distribuzione, mentre la distribuzione è intesa come la serie di applicazioni che circondano il kernel per renderlo utilizzabile	X
Il kernel è la distribuzione Linux	
Tutte le distribuzioni che utilizzano lo stesso kernel sono uguali	

4. Quale dei seguenti è un ambiente desktop in Linux?

Mint	
Elementary	
Zorin	
Gnome	X

5. Quale componente di un sistema operativo consente l'accesso all'hardware?

Driver	X
Shell	
Servizio	
Applicazione	

Risposte agli Esercizi Esplorativi

1. Se hai accesso alla Command Line, trova il rilascio corrente del kernel del tuo sistema.

```
$ uname -r
4.15.0-47-generic
```

2. Utilizzando il tuo motore di ricerca preferito, individua i provider di cloud pubblico a tua disposizione. Questi potrebbero includere AWS, Google Cloud, Rackspace e molti altri. Scegline uno e controlla quali sistemi operativi sono disponibili per l'installazione.

   AWS, per esempio, ti consente di installare molte distribuzioni Linux come Debian, Red Hat, SUSE o Ubuntu, come pure Windows.

4.2 Comprendere l'Hardware del Computer

Obiettivi LPI di riferimento

Linux Essentials version 1.6, Exam 010, Objective 4.2

Peso

2

Aree di Conoscenza Chiave

- Hardware

Di seguito è riportato un elenco parziale dei file, dei termini e dei comandi utilizzati

- Schede madri, processori, alimentatori, unità ottiche, periferiche
- Dischi rigidi, dischi a stato solido e partizioni, /dev/sd*
- Driver

4.2 Lezione 1

Certificazione:	Linux Essentials
Versione:	1.6
Argomento:	4 Il Sistema Operativo Linux
Obiettivo:	4.2 Comprendere l'Hardware del Computer
Lezione:	1 di 1

Introduzione

Senza hardware, il software non è altro che una forma di linguaggio. L'hardware elabora i comandi definiti dal software e fornisce meccanismi di archiviazione, input e output. Fondamentalmente, anche la tecnologia cloud è legata all'hardware.

In quanto sistema operativo, una delle responsabilità di Linux è dotare il software di interfacce per accedere all'hardware di sistema. La maggior parte delle specifiche di configurazione esula dallo scopo di questa lezione. Tuttavia, gli utenti devono spesso preoccuparsi delle prestazioni, della capacità e di altri fattori correlati all'hardware di sistema poiché influiscono sulla capacità del sistema stesso di supportare adeguatamente specifiche applicazioni. Questa lezione tratta l'hardware come elementi fisici separati utilizzando connettori e interfacce standard. Gli standard sono relativamente statici, ma la forma, le prestazioni e la capacità dell'hardware sono in continua evoluzione. Indipendentemente da come i cambiamenti possano nascondere le differenze fisiche, gli aspetti concettuali dell'hardware descritti in questa lezione rimangono ancora validi.

NOTE In vari punti di questa lezione verranno utilizzati esempi dalla Command Line per dimostrare come accedere alle informazioni sull'hardware. La maggior parte degli

> esempi proviene da un Raspberry Pi B+, ma vale per la maggior parte dei sistemi. Non è necessario comprendere questi comandi per capire la lezione.

Alimentatori

Tutti i componenti attivi in un computer richiedono elettricità per poter funzionare. Sfortunatamente la maggior parte delle fonti di elettricità non sono adeguate. L'hardware di un computer richiede tensioni specifiche con tolleranze relativamente strette, il che non è proprio quello che è disponibile dalla presa a muro a casa.

Gli alimentatori normalizzano le fonti di energia disponibili. I requisiti di tensione standard consentono ai produttori di creare componenti hardware che possono essere utilizzati nei sistemi in qualsiasi parte del mondo. Gli alimentatori dei computer desktop tendono a utilizzare come fonte di alimentazione l'elettricità delle prese a muro. Gli alimentatori dei server tendono ad avere una maggiore criticità; quindi sono spesso collegati a più fonti di alimentazione per garantire continuità operativa anche in caso di guasto di una fonte.

Il consumo di energia genera calore. Un calore eccessivo può far sì che i componenti del sistema funzionino più lentamente o addirittura si possano guastare. La maggior parte dei sistemi prevede una qualche forma di ventilazione per un raffreddamento più efficiente. Componenti come i processori spesso generano calore che il flusso d'aria da solo non è in grado di dissipare. Questi componenti caldi sono dotati di speciali alette note come dissipatori di calore che aiutano a dissipare il calore che generano. I dissipatori di calore hanno spesso una propria piccola ventola locale per garantire un flusso d'aria adeguato.

Scheda Madre

Tutto l'hardware di un sistema deve essere interconnesso. La scheda madre normalizza tale interconnessione utilizzando connettori e fattori di forma standardizzati. Fornisce inoltre supporto per la configurazione e le esigenze elettriche di tali connettori.

Esistono molte possibili configurazioni per una scheda madre. Prevedono processori e sistemi di memoria diversi e differenti combinazioni di connettori standardizzati. Inoltre hanno diverse dimensioni a seconda di dove sono contenute. La configurazione della scheda madre è del tutto trasparente agli utenti, i quali, forse, sono solo a conoscenza che possono collegare dispositivi esterni specifici. Gli amministratori di sistema di solito hanno a che fare con la configurazione della scheda madre quando devono identificare determinati dispositivi.

Quando viene accesa per la prima volta, è necessario configurare e inizializzare l'hardware specifico della scheda madre prima che il sistema possa funzionare. Le schede madri utilizzano programmi memorizzati nella memoria non volatile, nota come *firmware*, per gestire l'hardware

specifico della scheda madre. La forma originale del firmware della scheda madre era nota come BIOS (*Basic Input/Output System*). Oltre alle impostazioni della configurazione di base, il BIOS era principalmente responsabile dell'identificazione, del caricamento e del trasferimento delle operazioni a un sistema operativo come Linux. Con l'evoluzione dell'hardware il firmware si è espanso per supportare dischi più grandi, strumenti di diagnostica, interfacce grafiche, strumenti per la gestione della rete e altre funzionalità avanzate indipendenti da qualsiasi sistema operativo caricato. I primi tentativi di far avanzare il firmware oltre il BIOS di base erano spesso specifici per un produttore di schede madri. Intel ha definito uno standard per il firmware avanzato noto come EFI (*Extensible Firmware Interface*) e ha contribuito con EFI a un'organizzazione degli standard per creare UEFI (*Unified Extensible Firmware Interface*). Oggi, la maggior parte delle schede madri utilizza UEFI; è raro trovare BIOS ed EFI sui sistemi più recenti. Indipendentemente da ciò, la maggior parte delle persone si riferisce ancora al firmware della scheda madre utilizzando il termine BIOS.

Ci sono pochissime impostazioni del firmware realmente di interesse per gli utenti generici; quindi, in genere, solo i responsabili della configurazione hardware del sistema devono occuparsi del firmware e delle sue impostazioni. Una delle poche opzioni comunemente modificate è l'abilitazione delle estensioni di virtualizzazione delle moderne CPU.

Memoria

La memoria di sistema contiene i dati e il codice delle applicazioni attualmente in esecuzione. Quando si parla di memoria di un computer, la maggior parte delle persone si riferisce a questa memoria di sistema. Un altro termine comune con cui viene solitamente chiamata è l'acronimo RAM (*Random Access Memory*) o qualche variazione di tale acronimo. A volte vengono utilizzati anche riferimenti alla confezione fisica della memoria di sistema come, per esempio, DIMM, SIMM o DDR.

Fisicamente la memoria di sistema è solitamente confezionata in singoli moduli di circuiti stampati che si collegano alla scheda madre. I singoli moduli di memoria attualmente variano da 2 GB a 64 GB. Per la maggior parte delle applicazioni, 4 GB è la memoria di sistema minima che le persone dovrebbero tenere in considerazione. Per le singole postazioni di lavoro 16 GB sono in genere più che sufficienti. Tuttavia, anche 16 GB potrebbero essere pochi per gli utenti che eseguono giochi, video o applicazioni audio di fascia alta. I server richiedono spesso 128 GB o addirittura 256 GB di memoria per supportare in modo efficiente i carichi degli utenti.

Nella maggior parte dei casi Linux consente agli utenti di trattare la memoria di sistema come una *black box*. Viene avviata un'applicazione e Linux si occupa di allocare la memoria di sistema richiesta. Quando un'applicazione viene chiusa, Linux rilascia la memoria affinchè venga utilizzata da altre applicazioni. Ma cosa succede se un'applicazione richiede più memoria di sistema di quella disponibile? In questo caso, Linux sposta le applicazioni inattive dalla memoria

di sistema in una speciale area del disco, nota come spazio di swap. Linux sposta le applicazioni inattive dallo spazio di swap del disco alla memoria di sistema quando devono essere eseguite.

I sistemi senza un hardware video dedicato spesso utilizzano una parte della memoria di sistema (in genere 1 GB) come memoria del display video, riducendo la memoria di sistema effettiva. L'hardware video dedicato, in genere, ha una propria memoria separata che non è disponibile come memoria di sistema.

Esistono diversi modi per ottenere informazioni sulla memoria di sistema. Per un utente, i valori in genere più interessanti sono la quantità totale di memoria disponibile e in uso. Puoi ottenere delle informazioni eseguendo il comando `free` con il parametro `-m`, che visualizza l'output in megabyte:

```
$ free -m
              total        used        free      shared  buff/cache   available
Mem:            748          37          51          14         660         645
Swap:            99           0          99
```

La prima riga specifica la memoria totale disponibile per il sistema (`total`), la memoria in uso (`used`) e la memoria libera (`free`); la seconda riga mostra le stesse informazioni per lo spazio di swap. La memoria indicata come `shared` e `buff/cache` è attualmente usata per altre funzioni di sistema, sebbene la quantità indicata in `available` possa essere usata per le applicazioni.

Processori

La parola "processore" implica che qualcosa venga appunto processato. Nei computer la maggior parte di tale elaborazione si riferisce ai segnali elettrici. Tipicamente questi segnali vengono trattati come aventi uno dei valori binari, 1 o 0.

Quando le persone parlano di computer usano spesso la parola processore come sinonimo dell'acronimo CPU (*Central Processing Unit*), cosa non tecnicamente corretta. Ogni computer a uso generico ha una CPU che elabora i comandi binari specificati dal software. Quindi è comprensibile che le persone usino processore e CPU come sinonimi. Tuttavia, oltre a una CPU, i computer moderni spesso includono altri processori specifici per certe attività. Forse il processore aggiuntivo più noto è la GPU (*Graphical Processing Unit*). Pertanto, mentre una CPU è un processore, non tutti i processori sono CPU.

Per la maggior parte delle persone l'architettura della CPU fa riferimento alle istruzioni supportate dal processore. Sebbene Intel e AMD realizzino processori che supportano le stesse istruzioni, ha senso fare distinzione tra i fornitori poichè hanno differenze in termine di confezionamento, prestazioni e consumo energetico. Le distribuzioni software usano di solito

questi termini per specificare il set minimo di istruzioni richieste per il funzionamento:

i386

Fa riferimento al set di istruzioni a 32 bit associato a Intel 80386.

x86

In genere fa riferimento ai set di istruzioni a 32 bit associati ai successori del 80386 come 80486, 80586 e Pentium.

x64 / x86-64

Fa riferimento a processori che supportano sia le istruzioni a 32 bit che a 64 bit della famiglia x86.

AMD

Fa riferimento al supporto x86 dei processori AMD.

AMD64

Fa riferimento al supporto x64 dei processori AMD.

ARM

Fa riferimento a una CPU RISC (*Reduced Instruction Set Computer*) che non è basata sul set di istruzioni x86. È comunemente utilizzato da dispositivi integrati, mobili, tablet e a batteria. Una versione di Linux per ARM viene utilizzata dal Raspberry Pi.

Il file `/proc/cpuinfo` contiene informazioni dettagliate sul processore di un sistema. Purtroppo i dettagli non sono facilmente comprensibili dagli utenti comuni. Un risultato più generico può essere ottenuto con il comando `lscpu`. Ecco l'output di un Raspberry Pi B +:

```
$ lscpu
Architecture:          armv7l
Byte Order:            Little Endian
CPU(s):                4
On-line CPU(s) list:   0-3
Thread(s) per core:    1
Core(s) per socket:    4
Socket(s):             1
Model:                 4
Model name:            ARMv7 Processor rev 4 (v7l)
CPU max MHz:           1400.0000
CPU min MHz:           600.0000
BogoMIPS:              38.40
Flags:                 half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt vfpd32
```

```
lpae evtstrm crc32
```

Per la maggior parte delle persone, la miriade di fornitori, le famiglie di processori e i fattori di specifica rappresentano un'incredibile varietà di opzioni. Nonostante questo, ci sono diversi fattori associati a CPU e processori che anche gli utenti generici e gli amministratori spesso devono tenere in considerazione quando definiscono gli ambienti operativi:

Bit size

Per le CPU questo numero si riferisce sia alla dimensione nativa dei dati che gestisce, sia alla quantità di memoria cui può accedere. La maggior parte dei sistemi moderni è a 32 o 64 bit. Se un'applicazione deve accedere a più di 4 gigabyte di memoria, deve essere eseguita su un sistema a 64 bit, poiché 4 gigabyte è il numero massimo di indirizzi che può essere rappresentato utilizzando 32 bit. Inoltre, mentre le applicazioni a 32 bit possono in genere essere eseguite su sistemi a 64 bit, le applicazioni a 64 bit non possono essere eseguite su sistemi a 32 bit.

Clock speed

Spesso espressa in megahertz (MHz) o gigahertz (GHz), si riferisce alla velocità con cui un processore elabora le istruzioni. Ma la velocità del processore è solo uno dei fattori che influiscono sui tempi di risposta del sistema, sui tempi di attesa e sul *throughput*. Anche un utente multi-tasking molto impegnato raramente mantiene attiva la CPU di un comune PC desktop per più del due o tre per cento del tempo. Indipendentemente da questo, se si utilizzano frequentemente applicazioni ad alta intensità di calcolo che coinvolgono attività come la crittografia o il rendering video, la velocità della CPU potrebbe avere un impatto significativo sul *throughput* e sui tempi di attesa.

Cache

Le CPU richiedono un flusso costante di istruzioni e dati per funzionare. Il costo e il consumo energetico di una memoria di sistema multi-gigabyte, a cui si potrebbe accedere alla velocità di clock della CPU, sarebbero però proibitivi. La memoria cache della CPU è integrata nel chip della CPU per fornire un buffer ad alta velocità tra le CPU e la memoria di sistema. La cache è suddivisa in diversi livelli, comunemente indicati come L1, L2, L3 e persino L4. In generale, nel caso della cache, più ce ne sono meglio è.

Core

Un *core* si riferisce a una singola CPU. Oltre al *core* che rappresenta una CPU fisica, la *tecnologia Hyper-Threading* (HTT - Hyper-Threading Technology) consente a una singola CPU fisica di elaborare contemporaneamente più istruzioni agendo così virtualmente come più CPU fisiche. In genere, più *core* fisici vengono integrati in un singolo chip del processore fisico. Tuttavia, esistono schede madri che supportano più chip di processori fisici. In teoria, avere più

core per le attività di elaborazione potrebbe sembrare che porti sempre un migliore *throughput* del sistema. Sfortunatamente, le applicazioni desktop, in genere, mantengono le CPU occupate solo il due o tre per cento del tempo; quindi l'aggiunta di più CPU per lo più inattive è probabile che si traduca in un miglioramento minimo del *throughput*. L'uso di più *core* è più adatto per l'esecuzione di applicazioni scritte per avere più thread operativi indipendenti come il rendering di frame video, il rendering di pagine Web o ambienti con macchine virtuali multiutente.

Archiviazione

I dispositivi di archiviazione rappresentano un metodo per conservare programmi e dati. Le *unità a disco rigido* (HDD - Hard Disk Drives) e le *unità a stato solido* (SSD - Solid State Drives) sono i dispositivi di archiviazione più comuni per server e desktop. Vengono utilizzate anche pennette USB e dispositivi ottici come DVD, ma raramente come dispositivo principale.

Come suggerisce il nome, un'unità a disco rigido memorizza le informazioni su uno o più dischi rigidi fisici. I dischi fisici sono ricoperti da supporti magnetici per rendere possibile l'archiviazione. I dischi sono contenuti in una confezione sigillata poiché polvere, piccole particelle e persino impronte digitali potrebbero interferire con la capacità dell'HDD di leggere e scrivere sui supporti magnetici.

Gli SSD sono di fatto varianti più sofisticate di pennette USB con una capacità notevolmente maggiore. Gli SSD memorizzano le informazioni in microchip: quindi, non ci sono parti in movimento.

Sebbene le tecnologie alla base degli HDD e degli SSD siano diverse, esistono dei fattori di confronto molto importanti. La capacità degli HDD si basa sul ridimensionamento dei componenti fisici, mentre la capacità degli SSD dipende dal numero di microchip. Gli SSD costano per gigabyte da 3 a 10 volte di più di un HDD. Per la lettura o la scrittura, un HDD deve attendere che un punto su un disco ruoti in una posizione nota, mentre gli SSD sono ad accesso casuale. Le velocità di accesso degli SSD sono in genere da 3 a 5 volte superiori rispetto ai dispositivi HDD. Dal momento che non hanno parti mobili, gli SSD consumano meno energia e sono più affidabili degli HDD.

La capacità di archiviazione è in costante aumento per HDD e SSD. Oggigiorno sono in genere disponibili HDD da 5 terabyte e SSD da 1 terabyte. In ogni caso, una grande capacità di archiviazione non è sempre la cosa migliore. Quando un dispositivo di archiviazione si guasta, le informazioni in esso contenute non sono più disponibili. E, naturalmente, il backup richiede più tempo quando ci sono molte informazioni di cui fare il backup. Per le applicazioni che leggono e scrivono molti dati, la latenza e le prestazioni possono essere più importanti della capacità.

I sistemi moderni utilizzano interfacce SCSI (*Small Computer System Interface*) o SATA (*Serial AT*

Attachment) per connettersi ai dispositivi di archiviazione. Queste interfacce sono in genere supportate dai corrispondenti connettori sulla scheda madre. Il caricamento iniziale avviene da un dispositivo di archiviazione collegato alla scheda madre. Le impostazioni del firmware definiscono l'ordine in cui accedere ai dispositivi per questo caricamento iniziale.

I sistemi di archiviazione noti come RAID (*Redundant Array of Independent Disks*) sono una tipica implementazione per evitare la perdita di informazioni. Un *array* RAID è costituito da più dispositivi fisici che contengono copie duplicate di informazioni. Se un dispositivo si guasta, tutte le informazioni rimangono ancora disponibili. Ci si riferisce alle diverse configurazioni RAID fisiche con i numeri: 0, 1, 5, 6 e 10. Ogni definizione implica differenti dimensioni di archiviazione, caratteristiche di prestazione e modi per memorizzare dati ridondanti o *checksum* per il ripristino dei dati. Al di là di un piccolo *overhead* di configurazione a livello di amministratore, l'esistenza di un RAID è effettivamente trasparente agli utenti.

I dispositivi di archiviazione in genere leggono e scrivono i dati come blocchi di byte. Il comando `lsblk` può essere utilizzato per elencare i dispositivi a blocchi disponibili su un sistema. Il seguente esempio è stato eseguito su un Raspberry Pi utilizzando una scheda SD come dispositivo di archiviazione. I dettagli dell'output saranno trattati nelle sezioni *Partizioni* e *Driver e File di Dispositivo*:

```
$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
mmcblk0     179:0    0 29.7G  0 disk
+-mmcblk0p1 179:1    0 43.9M  0 part /boot
+-mmcblk0p2 179:2    0 29.7G  0 part /
```

Partitizioni

Un dispositivo di archiviazione è in realtà una lunga sequenza di posizioni di archiviazione. Il partizionamento è il meccanismo che indica a Linux se deve vedere queste posizioni di archiviazione come un'unica sequenza o più sequenze indipendenti. Ogni partizione viene trattata come se fosse un singolo dispositivo. Nella maggior parte dei casi le partizioni vengono create quando un sistema viene configurato per la prima volta. In caso fosse necessario apportare una modifica sono disponibili strumenti di amministrazione che gestiscono il partizionamento dei dispositivi.

Allora perchè è preferibile avere più partizioni? Alcuni esempi di utilizzo delle partizioni potrebbero essere la gestione dello spazio di archiviazione disponibile, l'isolamento dell'*overhead* di crittografia o il supporto di più file system. Le partizioni consentono di avere un singolo dispositivo di archiviazione che può essere avviato con diversi sistemi operativi.

Sebbene Linux possa riconoscere la sequenza di archiviazione di un dispositivo raw, questo non può essere utilizzato così com'è. Per utilizzare un dispositivo raw, è necessario formattarlo. La formattazione scrive un *file system* su un dispositivo e lo prepara per le operazioni sui file. Senza un file system non è possibile utilizzare un dispositivo per compiere operazioni sui file.

Gli utenti vedono le partizioni come se fossero dispositivi diversi: per questo è facile dimenticare che esiste un unico dispositivo fisico. In particolare, le operazioni da dispositivo a dispositivo, che sono in realtà operazioni da partizione a partizione, non avranno le prestazioni che ci si aspetta. Un singolo dispositivo è un meccanismo fisico con un un insieme di hardware per lettura/scrittura. Ma la cosa più importante è che non è possibile utilizzare le partizioni di un singolo dispositivo fisico per una progettazione a tolleranza di errore. Se il dispositivo si guasta, tutte le partizioni daranno errore; quindi non c'è tolleranza agli errori.

NOTE LVM (*Logical Volume Manager*) è una funzione software che consente agli amministratori di combinare singoli dischi e partizioni del disco e trattarli come se fossero un'unica unità.

Periferiche

I server e le *workstation* necessitano di una combinazione di CPU, memoria di sistema e dispositivi di archiviazione per poter funzionare. Ma questi componenti fondamentali non si interfacciano direttamente con il mondo esterno. Le periferiche sono i dispositivi che forniscono ai sistemi input, output e accesso al resto del mondo reale.

La maggior parte delle schede madri dispone di connettori esterni integrati e supporto firmware per interfacce periferiche *legacy* comuni per dispositivi come tastiera, mouse, audio, video e rete. Le schede madri più recenti in genere includono un connettore Ethernet per le reti, un connettore HDMI per le esigenze grafiche di base e uno o più connettori USB (*Universal Serial Bus*) per quasi tutto il resto. Esistono diverse versioni di USB con differenti velocità e caratteristiche fisiche. È comune avere più versioni di porte USB su una singola scheda madre.

Le schede madri possono anche avere uno o più slot di espansione che consentono agli utenti di aggiungere circuiti stampati speciali noti come schede di espansione per periferiche personalizzate, *legacy* e non standard. Le interfacce grafiche, audio e di rete sono schede di espansione molto comuni. Le schede di espansione supportano anche il RAID e interfacce *legacy* di formato speciale per connessioni seriali e parallele.

Le configurazioni *System on a Chip* (SoC) portano vantaggi in termini di consumo energetico, prestazioni, spazio e affidabilità rispetto alle configurazioni della scheda madre, poichè combinano processori, memoria di sistema, SSD e hardware di controllo delle periferiche in un unico pacchetto di circuiti integrati. Le periferiche supportate dalle configurazioni SoC sono

limitate dai componenti inclusi. Pertanto, le configurazioni SoC tendono a essere progettate per usi specifici. Telefoni, tablet e altri dispositivi portatili sono spesso basati sulla tecnologia SoC.

Alcuni sistemi sono già dotati di periferiche integrate. I laptop sono simili alle *workstation*, ma con periferiche integrate come schermo, tastiera e mouse predefiniti. I sistemi *all-in-one* sono simili ai laptop, ma richiedono mouse e tastiera. I controller di una scheda madre o un SoC sono spesso dotati di periferiche integrate appropriate progettate per un uso specifico.

Driver e File di Dispositivo

Finora, questa lezione ha presentato informazioni su processori, memoria, dischi, partizionamento, formattazione e periferiche. Ma chiedere a utenti generici di occuparsi dei dettagli specifici di ciascuno dei dispositivi nel loro sistema renderebbe questi sistemi praticamente inutilizzabili. Allo stesso modo, gli sviluppatori software dovrebbero modificare il loro codice per ogni dispositivo nuovo o modificato che devono supportare.

La soluzione a questo problema di "gestione dei dettagli" ci è data dai driver di un dispositivo. Questi accettano un insieme standard di richieste che poi traducono in attività di controllo appropriate per il dispositivo. I driver di un dispositivo consentono a te e alle applicazioni che esegui di leggere dal file `/home/carol/stuff` senza preoccuparti se quel file è su un disco rigido, un'unità a stato solido, una chiavetta di memoria, una memoria crittografata o su un qualche altro dispositivo.

I file dei dispositivi si trovano nella directory `/dev` e individuano i dispositivi fisici, le modalità di accesso ai dispositivi e i driver supportati. Per convenzione, nei sistemi moderni che utilizzano dispositivi di archiviazione basati su SCSI o SATA, il nome del file del relativo dispositivo inizia con il prefisso `sd`. Tale prefisso è seguito poi da una lettera come `a` o `b` che indica un dispositivo fisico. Dopo il prefisso e l'identificativo del dispositivo c'è un numero che indica una partizione all'interno del dispositivo fisico. Quindi, `/dev/sda` fa riferimento al primo dispositivo di archiviazione nella sua interezza, mentre `/dev/sda3` fa riferimento alla partizione 3 del primo dispositivo di archiviazione. Esiste una convenzione di nomenclatura per ogni tipo di dispositivo. Sebbene coprire tutte le possibili convenzioni di nomenclatura esuli dallo scopo di questa lezione, è importante ricordare che queste convenzioni sono fondamentali per l'amministrazione del sistema.

Sebbene trattare il contenuto della directory `/dev` esuli dallo scopo di questa lezione, può comunque essere istruttivo dare un'occhiata a una voce di un dispositivo di archiviazione. I file di dispositivo per le schede SD, in genere, utilizzano `mmcblk` come prefisso:

```
$ ls -l mmcblk*
brw-rw---- 1 root disk 179, 0 Jun 30 01:17 mmcblk0
```

```
brw-rw---- 1 root disk 179, 1 Jun 30 01:17 mmcblk0p1
brw-rw---- 1 root disk 179, 2 Jun 30 01:17 mmcblk0p2
```

I dettagli dell'elenco di un file di dispositivo sono diversi da quelli tipici dei file:

- A differenza di un file o di una directory, la prima lettera del campo dei permessi è `b`. Questo indica che i dati vengono letti e scritti nel dispositivo in blocchi anziché in singoli caratteri.
- Il campo della dimensione è costituito da due valori separati da una virgola anziché da un singolo valore. Il primo valore generalmente indica un driver particolare all'interno del kernel, mentre il secondo valore definisce uno specifico dispositivo gestito dal driver.
- Il nome del file utilizza un numero per il dispositivo fisico; quindi la convenzione di nomenclatura si adatta specificando il suffisso della partizione con una `p` seguita da un numero.

NOTE

Ogni dispositivo di sistema dovrebbe avere una voce in `/dev`. Poiché il contenuto della directory `/dev` viene creato durante l'installazione, spesso ci sono voci per tutti i driver e dispositivi possibili, anche se non esiste alcun dispositivo fisico.

Esercizi Guidati

1. Descrivi i seguenti termini:

Processore	
CPU	
GPU	

2. Se esegui principalmente applicazioni di editing video (un'attività ad alta intensità di calcolo), quali componenti e caratteristiche ti aspetti che abbiano un impatto sull'usabilità del sistema?

   Core della CPU - Velocità della CPU - Memoria di sistema disponibile - Sistema di archiviazione - GPU - Display video

3. Quale ti aspetti che sia il nome del file di dispositivo in `/dev` per la partizione 3 della terza unità SATA all'interno del sistema?

`sd3p3`	
`sdcp3`	
`sdc3`	

Esercizi Esplorativi

1. Esegui il comando `lsblk` sul tuo sistema e trova le informazioni richieste qui sotto. Se non hai a disposizione un sistema, considera l'elenco riportato da `lsblk -f` per il sistema Raspberry Pi menzionato precedentemente nella sezione "Archiviazione":

```
$ lsblk -f
NAME        FSTYPE LABEL  UUID                                 MOUNTPOINT
mmcblk0
+-mmcblk0p1 vfat   boot   9304-D9FD                            /boot
+-mmcblk0p2 ext4   rootfs 29075e46-f0d4-44e2-a9e7-55ac02d6e6cc /
```

   - Tipo e numero dei dispositivi
   - Struttura delle partizioni di ogni dispositivo
   - Tipo di file system e punto di montaggio per ogni partizione

Sommario

Un sistema è la somma dei suoi componenti. I diversi componenti influenzano costi, prestazioni e usabilità in modi diversi. Sebbene esistano configurazioni comuni per *workstation* e server, non esiste una configurazione migliore delle altre.

Risposte agli Esercizi Guidati

1. Descrivi i seguenti termini:

   Processore

   Un termine generale per qualsiasi tipo di processore. Viene spesso utilizzato in modo errato come sinonimo di CPU.

   CPU

   Un'unità centrale di elaborazione (è l'acronimo di Central Processing Unit). È un'unità di elaborazione che fornisce supporto per attività computazionali generiche.

   GPU

   Un'unità di elaborazione grafica (è l'acronimo di Graphical Processing Unit). È un'unità di elaborazione ottimizzata per supportare le attività che coinvolgono aspetti grafici.

2. Se esegui principalmente applicazioni di editing video (un'attività ad alta intensità di calcolo), quali componenti e caratteristiche ti aspetti che abbiano un impatto sull'usabilità del sistema?

   Core della CPU

   Sì. Più core supportano le attività contemporeanee di presentazione e rendering richieste dall'editing video.

   Velocità della CPU

   Sì. Il rendering video richiede una notevole quantità di attività di calcolo.

   Memoria di sistema disponibile

   Probabilmente. Il video non compresso utilizzato nell'editing è grande. I sistemi a uso generico spesso sono dotati di 8 gigabyte di memoria. 16 o addirittura 32 gigabyte di memoria consentono a un sistema di gestire più fotogrammi di video non compresso, rendendo le attività di editing più efficienti.

   Sistema di archiviazione

   Sì. I file video sono grandi. L'overhead delle unità SSD locali contribuisce a un trasferimento più efficiente. Unità di rete più lente sarebbero controproducenti.

   GPU

   No. La GPU influisce principalmente sulla presentazione del video renderizzato.

Display video

No. Il display video influisce principalmente sulla presentazione del video renderizzato.

3. Quale ti aspetti che sia il nome del file di dispositivo in `/dev` per la partizione 3 della terza unità SATA all'interno del sistema?

`sd3p3`	Non corretto. L'unità 3 dovrebbe essere `sdc` e non `sd3`
`sdcp3`	Non corretto. La partizione 3 dovrebbe essere `3` e non `p3`
`sdc3`	Corretto

Risposte agli Esercizi Esplorativi

1. Esegui il comando `lsblk` sul tuo sistema e trova le informazioni richieste qui sotto. Se non hai a disposizione un sistema, considera l'elenco riportato da `lsblk -f` per il sistema Raspberry Pi menzionato precedentemente nella sezione "Archiviazione":

```
$ lsblk -f
NAME        FSTYPE LABEL  UUID                                 MOUNTPOINT
mmcblk0
+-mmcblk0p1 vfat   boot   9304-D9FD                            /boot
+-mmcblk0p2 ext4   rootfs 29075e46-f0d4-44e2-a9e7-55ac02d6e6cc /
```

   Le risposte che seguono si basano sull'elenco riportato da `lsblk -f` per il sistema Raspberry Pi menzionato in precedenza . Le tue risposte potrebbero essere diverse:

   Tipo e numero dei dispositivi

   C'è un dispositivo: `mmcblk0`. Sappiamo che, per convenzione, `mmcblk` è una scheda di memoria SD.

   Struttura delle partizioni di ogni dispositivo

   Ci sono due partizioni: `mmcblk0p1` e `mmcblk0p2`.

   Tipo di file system e punto di montaggio per ogni partizione

   La partizione 1 utilizza il file system `vfat`. Viene utilizzata per avviare il sistema ed è montata come `/boot`. La partizione 2 utilizza il file system `ext4`. Viene utilizzata come file system principale ed è montata come `/`.

4.3 Dove Sono Memorizzati i Dati

Obiettivi LPI di riferimento

Linux Essentials version 1.6, Exam 010, Objective 4.3

Peso

3

Aree di Conoscenza Chiave

- Programmi e configurazione
- Processi
- Indirizzi di memoria
- Messaggistica di sistema
- Registrazione (Logging)

Di seguito è riportato un elenco parziale dei file, dei termini e dei comandi utilizzati

- `ps`, `top`, `free`
- `syslog`, `dmesg`
- `/etc/`, `/var/log/`
- `/boot/`, `/proc/`, `/dev/`, `/sys/`

4.3 Lezione 1

Certificazione:	Linux Essentials
Versione:	1.6
Argomento:	4 Il Sistema Operativo Linux
Obiettivo:	4.3 Dove Sono Memorizzati i Dati
Lezione:	1 di 2

Introduzione

Per un sistema operativo ogni cosa è un dato. Per Linux, ogni cosa è un file: programmi, file regolari, directory, dispositivi a blocchi (dischi rigidi, etc.), dispositivi a caratteri (console, etc.), processi del kernel, socket, partizioni, link, etc: la struttura delle directory di Linux , che inizia dalla *root* `/`, è un insieme di file contenenti dati. Il fatto che tutto sia considerato un file è una potente funzionalità di Linux in quanto consente di modificare praticamente ogni singolo aspetto del sistema.

In questa lezione discuteremo le diverse posizioni in cui vengono memorizzati i dati importanti, come stabilito dal Linux Filesystem Hierarchy Standard (FHS). Alcune di queste posizioni sono delle directory reali che memorizzano i dati in modo persistente sui dischi, mentre altre rappresentano pseudo-filesystem che vengono caricati in memoria e danno accesso ai dati del sottosistema del kernel come, per esempio, i processi in esecuzione, l'utilizzo della memoria, la configurazione hardware e così via. I dati memorizzati in queste directory virtuali vengono utilizzati da una serie di comandi che ci consentono di monitorarli e gestirli.

I Programmi e la Loro Configurazione

I dati importanti su un sistema Linux sono, senza dubbio, i programmi e i loro file di configurazione. I primi sono file eseguibili contenenti un'insieme di istruzioni che devono essere eseguite dal processore del computer, mentre i secondi sono solitamente documenti di testo che controllano il funzionamento di un programma. I file eseguibili possono essere file binari o file di testo. I file di testo eseguibili sono chiamati *script*. Anche i dati di configurazione, in Linux, sono tradizionalmente archiviati in file di testo, sebbene esistano vari stili di rappresentazione di questi dati.

Dove sono Memorizzati i File Binari

Come ogni altro file, i file eseguibili risiedono in directory fondamentalmente collegate a `/`. Nello specifico, i programmi sono distribuiti su una struttura a tre livelli: il primo livello (`/`) include i programmi che potrebbero essere necessari in modalità utente singolo; il secondo livello (`/usr`) contiene la maggior parte dei programmi multiutente; il terzo livello (`/usr/local`) viene utilizzato per memorizzare il software che non è fornito dalla distribuzione ed è stato compilato localmente.

In genere, i programmi si trovano in queste posizioni:

`/sbin`
: Contiene i binari essenziali per l'amministrazione del sistema, come `parted` o `ip`.

`/bin`
: Contiene i binari essenziali per tutti gli utenti come `ls`, `mv`, o `mkdir`.

`/usr/sbin`
: Memorizza i binari per l'amministrazione del sistema, come `deluser` o `groupadd`.

`/usr/bin`
: Include la maggior parte dei file eseguibili, come `free`, `pstree`, `sudo` o `man`, che possono essere utilizzati da tutti gli utenti.

`/usr/local/sbin`
: Viene utilizzato per memorizzare i programmi installati localmente per l'amministrazione del sistema che non sono gestiti dal gestore dei pacchetti del sistema.

`/usr/local/bin`
: Ha lo stesso scopo di `/usr/local/sbin`, ma per i normali programmi utente.

Recentemente alcune distribuzioni hanno iniziato a sostituire `/bin` e `/sbin` con collegamenti simbolici a `/usr/bin` e `/usr/sbin`.

NOTE La directory `/opt` è talvolta usata per memorizzare applicativi opzionali di terze parti.

Oltre a queste directory, gli utenti ordinari possono avere i propri programmi in:

- `/home/$USER/bin`
- `/home/$USER/.local/bin`

TIP Puoi scoprire da quali directory puoi eseguire i binari facendo riferimento alla variabile `PATH` con il comando `echo $PATH`. Per ulteriori informazioni su `PATH`, rivedi le lezioni sulle variabili e sulla personalizzazione della shell.

Possiamo trovare la posizione dei programmi con il comando `which`:

```
$ which git
/usr/bin/git
```

Dove sono Memorizzati i File di Configurazione

La Directory `/etc`

Agli albori di Unix c'era una cartella per ogni tipo di dato, come, per esempio, `/bin` per i binari e `/boot` per il (i) kernel. Tuttavia, `/etc` (per *et cetera*) è stata creata come directory generica per memorizzare tutti i file che non appartenevano alle altre categorie. La maggior parte di questi file erano file di configurazione. Con il passare del tempo sono stati aggiunti sempre più file di configurazione e così `/etc` è diventata la cartella principale per i file di configurazione dei programmi. Come detto sopra, un file di configurazione di solito è un file locale di testo normale (al contrario di un file binario) che controlla il funzionamento di un programma.

In `/etc` troviamo diversi modelli per i nomi dei file di configurazione:

- File con una estensione *ad hoc* o nessuna estensione, come per esempio:

  `group`
  : Database dei gruppi del sistema.

  `hostname`
  : Nome del computer host.

`hosts`

Elenco di indirizzi IP e relative traduzioni in nome host.

`passwd`

Database degli utenti del sistema — composto da sette campi separati da due punti che forniscono informazioni sull'utente.

`profile`

File di configurazione a livello di sistema per Bash.

`shadow`

File crittografato per le password degli utenti.

- File di inizializzazione che terminano con `rc`:

`bash.bashrc`

File `.bashrc` a livello di sistema per shell bash interattive.

`nanorc`

File di inizializzazione di esempio per GNU nano (un semplice editor di testo normalmente incluso in qualsiasi distribuzione).

- File che terminano con `.conf`:

`resolv.conf`

File di configurazione per il resolver, che fornisce l'accesso al DNS (Internet Domain Name System).

`sysctl.conf`

File di configurazione per impostare le variabili di sistema per il kernel.

- Directory con suffisso `.d`:

Alcuni programmi con un unico file di configurazione (`*.conf` o altro) si sono evoluti per avere una directory dedicata `*.d` che aiuta a costruire configurazioni modulari e più robuste. Per esempio, per configurare *logrotate*, troverai `logrotate.conf`, ma anche la directory `logrotate.d`.

Questo approccio è utile nei casi in cui applicazioni diverse richiedano configurazioni per lo stesso specifico servizio. Se, per esempio, un pacchetto del server web contiene una configurazione di logrotate, questa configurazione può essere inserita in un file dedicato nella

directory `logrotate.d`. Questo file può essere aggiornato dal pacchetto del server web senza interferire con la restante configurazione di logrotate. Allo stesso modo, i pacchetti possono aggiungere attività specifiche inserendo i file nella directory `/etc/cron.d` invece di modificare `/etc/crontab`.

In Debian e nelle sue derivate tale approccio è stato applicato all'elenco delle fonti attendibili lette dallo strumento di gestione dei pacchetti `apt`: a parte il classico `/etc/apt/sources.list`, ora troviamo la directory `/etc/apt/sources.list.d`:

```
$ ls /etc/apt/sources*
/etc/apt/sources.list
/etc/apt/sources.list.d:
```

File di Configurazione nella `HOME` (Dotfile)

A livello utente i programmi memorizzano le proprie configurazioni e le proprie impostazioni in file nascosti all'interno della directory home dell'utente (rappresentata anche da `~`). Ricorda che i file nascosti iniziano con un punto (`.`) - da qui il loro nome: *dotfile*.

Alcuni di questi dotfile sono script Bash che personalizzano la sessione di shell dell'utente e vengono letti non appena l'utente accede al sistema:

`.bash_history`
: Memorizza la cronologia della Command Line.

`.bash_logout`
: Contiene i comandi da eseguire quando si esce dalla shell di login.

`.bashrc`
: Script di inizializzazione di Bash per shell non di login.

`.profile`
: Script di inizializzazione di Bash per shell di login.

> **NOTE** Fai riferimento alla lezione "Nozioni di Base sulla Command Line" per saperne di più su Bash e sui suoi file di inizializzazione.

Altri file di configurazione relativi a programmi specifici dell'utente vengono letti quando il corrispondente programma viene avviato: `.gitconfig`, `.emacs.d`, `.ssh`, etc.

Il kernel Linux

Prima che qualsiasi processo possa essere eseguito, il kernel deve essere caricato in un'area di memoria protetta. Dopodiché, il processo con PID `1` (oggigiorno il più delle volte `systemd`) avvia la catena di processi, vale a dire un processo ne avvia un altro (o altri) e così via. Una volta che i processi sono attivi, il kernel Linux è responsabile dell'allocazione delle risorse (tastiera, mouse, dischi, memoria, interfacce di rete, etc.).

NOTE

Prima dell'avvento di `systemd`, `/sbin/init` era sempre il primo processo in un sistema Linux come parte del gestore di sistema *System V Init*. Infatti, al momento puoi ancora trovare `/sbin/init`, ma come collegamento a `/lib/systemd/systemd`.

Dove sono Memorizzati i File del Kernel: `/boot`

Il kernel risiede in `/boot` assieme ad altri file relativi all'avvio del sistema. La maggior parte di questi file include nel nome ciò che compone il numero di versione del kernel (kernel version, major revision, minor revision e patch number).

La directory `/boot` include i seguenti tipi di file, i cui nomi corrispondono alla relativa versione del kernel:

`config-4.9.0-9-amd64`

Impostazioni di configurazione del kernel come, per esempio, opzioni e moduli che sono stati compilati assieme al kernel.

`initrd.img-4.9.0-9-amd64`

Immagine del disco RAM iniziale (Initial RAM disk) che aiuta il processo di avvio caricando un filesystem di root temporaneo in memoria.

`System-map-4.9.0-9-amd64`

Il file `System-map` (su alcuni sistemi presente come `System.map`) contiene le posizioni degli indirizzi di memoria per i nomi dei simboli del kernel. Ogni volta che un kernel viene ricostruito, il contenuto del file cambia poiché le posizioni di memoria potrebbero essere diverse. Il kernel utilizza questo file per cercare le posizioni degli indirizzi di memoria per un particolare simbolo del kernel, o viceversa.

`vmlinuz-4.9.0-9-amd64`

Il kernel vero e proprio in un formato compresso autoestraente che consente di risparmiare spazio (da qui la `z` in `vmlinuz`; `vm` sta per memoria virtuale - ha iniziato a essere utilizzato quando il kernel ha iniziato a supportare la memoria virtuale).

`grub`

Directory di configurazione per il bootloader `grub2`.

TIP

Poiché è una criticità del sistema operativo, vengono conservati in `/boot` più di un kernel con i relativi file associati nel caso in cui quello predefinito dia errore e si debba ripiegare su una versione precedente così da essere almeno in grado di avviare e riparare il sistema.

La Directory `/proc`

La directory `/proc` è uno dei così detti filesystem virtuali o pseudo-filesystem dal momento che il suo contenuto non è scritto sul disco, ma caricato in memoria. Viene popolato dinamicamente ogni volta che il computer si avvia e riflette costantemente lo stato corrente del sistema. `/proc` contiene informazioni su:

- Processi in esecuzione
- Configurazione del kernel
- Hardware di sistema

Oltre a tutti i dati relativi ai processi che vedremo nella prossima lezione, questa directory memorizza anche file con informazioni sull'hardware di sistema e le impostazioni di configurazione del kernel. Alcuni di questi file sono:

`/proc/cpuinfo`

Memorizza informazioni sulla CPU del sistema:

```
$ cat /proc/cpuinfo
processor   : 0
vendor_id   : GenuineIntel
cpu family  : 6
model       : 158
model name  : Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz
stepping    : 10
cpu MHz     : 3696.000
cache size  : 12288 KB
(...)
```

`/proc/cmdline`

Memorizza le stringhe passate al kernel all'avvio del sistema:

```
$ cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64 root=UUID=5216e1e4-ae0e-441f-b8f5-8061c0034c74 ro
quiet
```

`/proc/modules`

Mostra l'elenco dei moduli caricati nel kernel:

```
$ cat /proc/modules
nls_utf8 16384 1 - Live 0xffffffffc0644000
isofs 40960 1 - Live 0xffffffffc0635000
udf 90112 0 - Live 0xffffffffc061e000
crc_itu_t 16384 1 udf, Live 0xffffffffc04be000
fuse 98304 3 - Live 0xffffffffc0605000
vboxsf 45056 0 - Live 0xffffffffc05f9000 (O)
joydev 20480 0 - Live 0xffffffffc056e000
vboxguest 327680 5 vboxsf, Live 0xffffffffc05a8000 (O)
hid_generic 16384 0 - Live 0xffffffffc0569000
(...)
```

La Directory `/proc/sys`

Questa directory contiene le impostazioni di configurazione del kernel sotto forma di file classificati in categorie rappresentate dalle diverse sottodirectory:

```
$ ls /proc/sys
abi  debug  dev  fs  kernel  net  user  vm
```

La maggior parte di questi file funziona come un interruttore e, quindi, contiene solamente uno dei due possibili valori: `0` o `1` ("acceso" o "spento"). Per esempio:

`/proc/sys/net/ipv4/ip_forward`

Il valore che abilita o disabilita la nostra macchina a funzionare da router (e quindi a essere in grado di inoltrare pacchetti):

```
$ cat /proc/sys/net/ipv4/ip_forward
0
```

Esistono tuttavia alcune eccezioni:

`/proc/sys/kernel/pid_max`

Il PID massimo consentito:

```
$ cat /proc/sys/kernel/pid_max
32768
```

WARNING Presta particolare attenzione quando modifichi le impostazioni del kernel poiché un valore errato potrebbe causare instabilità di sistema.

Dispositivi Hardware

Ricorda, in Linux "ogni cosa è un file". Questo implica che le informazioni sul dispositivo hardware e le impostazioni di configurazione del kernel siano tutte memorizzate in file speciali che risiedono in directory virtuali.

La Directory `/dev`

La directory `/dev`, che sta per *device*, contiene i file di dispositivo (o nodi) per tutti i dispositivi hardware collegati. Questi file di dispositivo vengono utilizzati come interfaccia tra i dispositivi e i processi che li utilizzano. Ogni file di dispositivo rientra in una delle due seguenti categorie:

Dispositivi a blocchi

Sono quelli in cui i dati vengono letti e scritti in blocchi che possono essere indirizzati individualmente. Alcuni esempi sono: i dischi rigidi (e le loro partizioni, come `/dev/sda1`), le unità flash USB, CD, DVD, etc.

Dispositivi a caratteri

Sono quelli in cui i dati vengono letti e scritti in modo sequenziale, un carattere alla volta. Alcuni esempi sono: le tastiere, la console di testo (`/dev/console`), le porte seriali (come `/dev/ttyS0` e così via), etc.

Quando si elencano i file di dispositivo è necessario assicurarsi di utilizzare `ls` con l'opzione `-l` per fare distinzione tra le due categorie. Possiamo, per esempio, controllare i dischi rigidi e le partizioni:

```
# ls -l /dev/sd*
brw-rw---- 1 root disk 8, 0 may 25 17:02 /dev/sda
brw-rw---- 1 root disk 8, 1 may 25 17:02 /dev/sda1
brw-rw---- 1 root disk 8, 2 may 25 17:02 /dev/sda2
```

```
(...)
```

O i terminali seriali (TeleTYpewriter):

```
# ls -l /dev/tty*
crw-rw-rw- 1 root tty     5,  0 may 25 17:26 /dev/tty
crw--w---- 1 root tty     4,  0 may 25 17:26 /dev/tty0
crw--w---- 1 root tty     4,  1 may 25 17:26 /dev/tty1
(...)
```

Nota che il primo carattere è `b` per i dispositivi a blocchi e `c` per i dispositivi a caratteri.

TIP

L'asterisco ("*") è un carattere di *globbing* che significa 0 o più caratteri. Pertanto, è estremamente importante nei comandi `ls -l /dev/sd*` e `ls -l /dev/tty*` sopra riportati. Per saperne di più su questi caratteri speciali, fai riferimento alla lezione sul globbing.

Inoltre, `/dev` include alcuni file speciali che sono abbastanza utili per diversi scopi di programmazione:

`/dev/zero`
: Fornisce tanti caratteri null quanti sono quelli richiesti.

`/dev/null`
: Chiamato anche *cestino per i bit*, o in inglese *bit bucket*. Elimina tutte le informazioni che gli vengono inviate.

`/dev/urandom`
: Genera numeri pseudo-casuali.

La Directory `/sys`

Il *filesystem sys* (`sysfs`) è montato su `/sys`. È stato introdotto con il kernel 2.6 e ha apportato un grande miglioramento per `/proc/sys`.

I processi devono interagire con i dispositivi in `/dev` e quindi il kernel ha bisogno di una directory che contenga informazioni su questi dispositivi hardware. Questa directory è `/sys` e i suoi dati sono organizzati in categorie. Per esempio, per controllare l'indirizzo MAC della scheda di rete (`enp0s3`), è necessario eseguire il comando `cat` sul seguente file:

```
$ cat /sys/class/net/enp0s3/address
08:00:27:02:b2:74
```

Memoria e Tipi di Memoria

Affinché un programma inizi a funzionare, deve essere caricato in memoria. In generale, quando parliamo di memoria ci riferiamo alla *Random Access Memory* (RAM) che, rispetto ai dischi rigidi meccanici, ha il vantaggio di essere molto più veloce. Di contro, è volatile (cioè, una volta spento il computer, i dati scompaiono).

Fermo restando quanto detto sopra, quando si tratta di memoria possiamo distinguerne in un sistema Linux due tipi principali:

Memoria fisica

Conosciuta anche come *RAM*, si presenta sotto forma di chip costituiti da circuiti integrati contenenti milioni di transistor e condensatori che, a loro volta, formano celle di memoria (il componente base della memoria del computer). Ciascuna di queste celle ha un codice esadecimale associato, un indirizzo di memoria, a cui è possibile fare riferimento quando necessario.

Swap

Conosciuta anche come *spazio di swap*, è la porzione di memoria virtuale che si trova sul disco rigido e viene utilizzata quando non c'è più RAM disponibile.

D'altra parte, esiste il concetto di *memoria virtuale* che è un'astrazione della quantità totale di memoria di indirizzamento utilizzabile (RAM, ma anche spazio su disco) come viene vista dalle applicazioni.

Il comando `free` analizza `/proc/meminfo` e mostra la quantità di memoria libera e utilizzata nel sistema in modo molto chiaro:

```
$ free
              total        used        free      shared  buff/cache   available
Mem:        4050960     1474960     1482260       96900     1093740     2246372
Swap:       4192252           0     4192252
```

Spieghiamo le diverse colonne:

`total`

Quantità totale di memoria fisica e di swap installata.

`used`
: Quantità di memoria fisica e di swap attualmente in uso.

`free`
: Quantità di memoria fisica e di swap attualmente non in uso.

`shared`
: Quantità di memoria fisica usata — principalmente — da `tmpfs`.

`buff/cache`
: Quantità di memoria fisica attualmente utilizzata dai buffer del kernel, dalla cache e dagli slab.

`available`
: Stima della quantità di memoria fisica disponibile per i nuovi processi.

Per impostazione predefinita, `free` mostra i valori in *kibibyte*, ma consente, grazie a una varietà di opzioni, di visualizzare i risultati in diverse unità di misura. Alcune di queste opzioni sono:

`-b`
: Bytes.

`-m`
: Mebibytes.

`-g`
: Gibibytes.

`-h`
: Formato leggibile dall'uomo.

Il formato mostrato tramite l'opzione `-h` è sempre comodo da leggere:

```
$ free -h
               total        used        free      shared  buff/cache   available
Mem:           3,9G        1,4G        1,5G         75M        1,0G        2,2G
Swap:          4,0G          0B        4,0G
```

NOTE Un kibibyte (KiB) equivale a 1024 byte, mentre un kilobyte (KB) equivale a 1000 byte. Lo stesso vale, rispettivamente, per mebibyte, gibibyte, etc.

Esercizi Guidati

1. Usa il comando `which` per scoprire la posizione dei seguenti programmi e completare la tabella:

Programma	**Comando `which`**	**Percorso dell'eseguibile (output)**	**L'utente ha bisogno dei privilegi di `root`?**
`swapon`			
`kill`			
`cut`			
`usermod`			
`cron`			
`ps`			

2. Dove si trovano i seguenti file?

File	`/etc`	`~`
`.bashrc`		
`bash.bashrc`		
`passwd`		
`.profile`		
`resolv.conf`		
`sysctl.conf`		

3. Spiega il significato degli elementi numerici del file del kernel `vmlinuz-4.15.0-50-generic` che si trova in `/boot`:

Elemento numerico	**Significato**
`4`	
`15`	
`0`	
`50`	

4. Quale comando puoi usare per elencare tutti i dischi rigidi e le partizioni in `/dev`?

Esercizi Esplorativi

1. I file di dispositivo per i dischi rigidi sono rappresentati in base ai controller che usano: abbiamo visto `/dev/sd*` per i dischi che utilizzano SCSI (Small Computer System Interface) e SATA (Serial Advanced Technology Attachment), ma
   - Come erano rappresentati i vecchi dischi IDE (Integrated Drive Electronics)?
   - E i moderni dischi NVMe (Non-Volatile Memory Express)?

2. Dai un'occhiata al file `/proc/meminfo`. Confronta il contenuto di questo file con l'output del comando `free` e identifica quale voce di `/proc/meminfo` corrisponde ai seguenti campi nell'output di `free`:

Output di `free`	**Campo di** `/proc/meminfo`
`total`	
`free`	
`shared`	
`buff/cache`	
`available`	

Sommario

In questa lezione hai imparato la posizione dei programmi e dei loro file di configurazione in un sistema Linux. Le cose più importanti da ricordare sono:

- Fondamentalmente, i programmi si trovano in una struttura di directory a tre livelli: `/`, `/usr` e `/usr/local`. Ciascuno di questi livelli può contenere le directory `bin` e `sbin`;
- I file di configurazione sono memorizzati in `/etc` e `~`;
- I dotfile sono file nascosti che iniziano con un punto (`.`).

Abbiamo anche discusso del kernel Linux. Le cose più importanti sono:

- Per Linux, ogni cosa è un file;
- Il kernel Linux si trova in `/boot` assieme agli altri file relativi all'avvio;
- Affinchè i processi possano essere eseguiti, il kernel deve prima essere caricato in un'area di memoria protetta;
- Il compito del kernel è allocare le risorse di sistema per i processi;
- Il filesystem virtuale (o pseudo filesystem) `/proc` memorizza, in modo volatile, importanti dati del kernel e del sistema.

Allo stesso modo, abbiamo esplorato i dispositivi hardware e appreso quanto segue:

- La directory `/dev` memorizza file speciali (o nodi) per tutti i dispositivi hardware collegati: *dispositivi a blocchi* o *dispositivi a caratteri*. I primi trasferiscono i dati in blocchi; i secondi, un carattere alla volta;
- La directory `/dev` contiene anche altri file speciali come `/dev/zero`, `/dev/null` o `/dev/urandom`;
- La directory `/sys` memorizza le informazioni sui dispositivi hardware in base a una suddivisione per categorie.

Infine, abbiamo affrontato la memoria. Ecco ciò che abbiamo imparato:

- Un programma viene eseguito quando viene caricato in memoria;
- Cos'è la RAM (Random Access Memory);
- Cos'è lo Swap;
- Come visualizzare l'utilizzo della memoria.

Comandi utilizzati in questa lezione:

`cat`

Concatena/stampa il contenuto di un file.

`free`

Visualizza la quantità di memoria libera e utilizzata nel sistema.

`ls`

Elenca il contenuto di una directory.

`which`

Mostra la posizione di un programma.

Risposte agli Esercizi Guidati

1. Usa il comando `which` per scoprire la posizione dei seguenti programmi e completare la tabella:

Programma	**Comando `which`**	**Percorso dell'eseguibile (output)**	**L'utente ha bisogno dei privilegi di `root`?**
`swapon`	`which swapon`	`/sbin/swapon`	Si
`kill`	`which kill`	`/bin/kill`	No
`cut`	`which cut`	`/usr/bin/cut`	No
`usermod`	`which usermod`	`/usr/sbin/usermod`	Si
`cron`	`which cron`	`/usr/sbin/cron`	Si
`ps`	`which ps`	`/bin/ps`	No

2. Dove si trovano i seguenti file?

File	`/etc`	`~`
`.bashrc`		X
`bash.bashrc`	X	
`passwd`	X	
`.profile`		X
`resolv.conf`	X	
`sysctl.conf`	X	

3. Spiega il significato degli elementi numerici del file del kernel `vmlinuz-4.15.0-50-generic` che si trova in `/boot`:

Elemento numerico	**Significato**
`4`	Kernel version
`15`	Major revision
`0`	Minor revision
`50`	Patch number

4. Quale comando puoi usare per elencare tutti i dischi rigidi e le partizioni in `/dev`?

   `ls /dev/sd*`

Risposte agli Esercizi Esplorativi

1. I file di dispositivo per i dischi rigidi sono rappresentati in base ai controller che usano: abbiamo visto `/dev/sd*` per i dischi che utilizzano SCSI (Small Computer System Interface) e SATA (Serial Advanced Technology Attachment), ma
   - Come erano rappresentati i vecchi dischi IDE (Integrated Drive Electronics)?

     `/dev/hd*`

   - E i moderni dischi NVMe (Non-Volatile Memory Express)?

     `/dev/nvme*`

2. Dai un'occhiata al file `/proc/meminfo`. Confronta il contenuto di questo file con l'output del comando `free` e identifica quale voce di `/proc/meminfo` corrisponde ai seguenti campi nell'output di `free`:

Output di `free`	**Campo di** `/proc/meminfo`
`total`	`MemTotal` / `SwapTotal`
`free`	`MemFree` / `SwapFree`
`shared`	`Shmem`
`buff/cache`	`Buffers`, `Cached` e `SReclaimable`
`available`	`MemAvailable`

4.3 Lezione 2

Certificazione:	Linux Essentials
Versione:	1.6
Argomento:	4 Il Sistema Operativo Linux
Obiettivo:	4.3 Dove Sono Memorizzati i Dati
Lezione:	2 di 2

Introduzione

Dopo aver esplorato i programmi e i loro file di configurazione, in questa lezione impareremo come i comandi vengano eseguiti come processi. Allo stesso modo, ci occuperemo dei messaggi di sistema, dell'uso del *kernel ring buffer* e di come l'avvento di `systemd` e del suo demone di journal, `journald`, abbiano cambiato il tradizionale modo di trattare i log di sistema.

Processi

Ogni volta che un utente lancia un comando, viene eseguito un programma e vengono generati uno o più processi.

I processi sono organizzati gerarchicamente. Dopo che il kernel è stato caricato in memoria durante l'avvio, viene avviato il primo processo che, a sua volta, avvia altri processi che, a loro volta, possono avviare altri processi. Ogni processo ha un identificativo univoco (`PID`) e un identificativo del processo padre (`PPID`). Questi sono numeri interi positivi assegnati in ordine sequenziale.

Esplorare i Processi in Modo Dinamico: `top`

Puoi ottenere un elenco dinamico di tutti i processi in esecuzione con il comando `top`:

```
$ top

top - 11:10:29 up  2:21,  1 user,  load average: 0,11, 0,20, 0,14
Tasks:  73 total,   1 running,  72 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0,0 us,  0,3 sy,  0,0 ni, 99,7 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
KiB Mem :  1020332 total,   909492 free,    38796 used,    72044 buff/cache
KiB Swap:  1046524 total,  1046524 free,        0 used.   873264 avail Mem

   PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
   436 carol     20   0   42696   3624   3060 R  0,7  0,4   0:00.30 top
     4 root      20   0       0      0      0 S  0,3  0,0   0:00.12 kworker/0:0
   399 root      20   0   95204   6748   5780 S  0,3  0,7   0:00.22 sshd
     1 root      20   0   56872   6596   5208 S  0,0  0,6   0:01.29 systemd
     2 root      20   0       0      0      0 S  0,0  0,0   0:00.00 kthreadd
     3 root      20   0       0      0      0 S  0,0  0,0   0:00.02 ksoftirqd/0
     5 root       0 -20       0      0      0 S  0,0  0,0   0:00.00 kworker/0:0H
     6 root      20   0       0      0      0 S  0,0  0,0   0:00.00 kworker/u2:0
     7 root      20   0       0      0      0 S  0,0  0,0   0:00.08 rcu_sched
     8 root      20   0       0      0      0 S  0,0  0,0   0:00.00 rcu_bh
     9 root      rt   0       0      0      0 S  0,0  0,0   0:00.00 migration/0
    10 root       0 -20       0      0      0 S  0,0  0,0   0:00.00 lru-add-drain
    (...)
```

Come mostrato nell'esempio sopra riportato, `top` fornisce anche informazioni sulla memoria e sul consumo di CPU dell'intero sistema così pure per ogni processo.

`top` consente all'utente alcune interazioni.

Per impostazione predefinita, l'output viene ordinato, in ordine descrescente, in base alla percentuale del tempo di CPU utilizzato da ciascun processo. Questo comportamento può essere modificato premendo i seguenti tasti all'interno di `top`:

`M`

Ordina per utilizzo della memoria.

`N`

Ordina in base al numero di ID di processo.

T

Ordina per tempo di esecuzione.

P

Ordina per percentuale di utilizzo della CPU.

Per cambiare ordinamento decrescente/crescente, basta premere `R`.

TIP Una versione più carina e più facile da utilizzare di `top` è `htop`. Un'altra alternativa, forse più esaustiva, è `atop`. Se non sono già installati sul tuo sistema, ti invito a utilizzare il tuo gestore di pacchetti per installarli e provarli.

Un'Istantanea dei Processi: `ps`

Un altro comando molto utile per ottenere informazioni sui processi è `ps`. Mentre `top` fornisce informazioni dinamiche, quelle di `ps` sono statiche.

Se invocato senza opzioni, l'output di `ps` è abbastanza soddisfacente e si riferisce solo ai processi collegati alla shell corrente:

```
$ ps
  PID TTY          TIME CMD
 2318 pts/0    00:00:00 bash
 2443 pts/0    00:00:00 ps
```

Le informazioni visualizzate includono l'identificativo di processo (`PID`), il terminale in cui viene eseguito il processo (`TTY`), il tempo di CPU utilizzato dal processo (`TIME`) e il comando che ha avviato il processo (`CMD`).

Un'opzione utile per `ps` è `-f` che mostra l'elenco in formato completo:

```
$ ps -f
UID        PID  PPID  C STIME TTY          TIME CMD
carol     2318  1682  0 08:38 pts/1    00:00:00 bash
carol     2443  2318  0 08:46 pts/1    00:00:00 ps -f
```

In combinazione con altre opzioni, `-f` mostra la relazione tra i processi "padre" e "figlio":

```
$ ps -uf
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
```

```
carol       2318  0.0  0.1  21336  5140 pts/1    Ss   08:38   0:00 bash
carol       2492  0.0  0.0  38304  3332 pts/1    R+   08:51   0:00  \_ ps -uf
carol       1780  0.0  0.1  21440  5412 pts/0    Ss   08:28   0:00 bash
carol       2291  0.0  0.7 305352 28736 pts/0    Sl+  08:35   0:00  \_ emacs index.en.adoc
-nw
(...)
```

Allo stesso modo, `ps` può mostrare la percentuale di memoria utilizzata quando viene invocato con l'opzione `-v`:

```
$ ps -v
  PID TTY      STAT   TIME  MAJFL   TRS   DRS   RSS %MEM COMMAND
 1163 tty2     Ssl+   0:00      1    67 201224 5576  0.1 /usr/lib/gdm3/gdm-x-session (...)
 (...)
```

NOTE Un altro comando, carino dal punto di vista estetico, che mostra la gerarchia dei processi è `pstree` incluso in tutte le principali distribuzioni.

Informazioni sui Processi nella Directory `/proc`

Abbiamo già visto il filesystem `/proc`. `/proc` include una sottodirectory numerata per ogni processo in esecuzione nel sistema (il numero è il `PID` del processo):

```
carol@debian:~# ls /proc
1    108  13   17   21   27   354  41   665  8    9
10   109  14   173  22   28   355  42   7    804  915
103  11   140  18   23   29   356  428  749  810  918
104  111  148  181  24   3    367  432  75   811
105  112  149  19   244  349  370  433  768  83
106  115  15   195  25   350  371  5    797  838
107  12   16   2    26   353  404  507  798  899
(...)
```

Pertanto, tutte le informazioni su un particolare processo sono incluse nella sua directory. Elenchiamo il contenuto del primo processo, ovvero quello il cui `PID` è `1` (l'output è stato troncato per una migliore leggibilità):

```
# ls /proc/1/
attr        cmdline          environ  io        mem       ns
autogroup   comm             exe      limits    mountinfo numa_maps
```

```
auxv        coredump_filter  fd       loginuid  mounts    oom_adj
...
```

Puoi controllare, per esempio, l'eseguibile del processo:

```
# cat /proc/1/cmdline; echo
/sbin/init
```

Come puoi vedere il binario che ha avviato la gerarchia dei processi è `/sbin/init`.

NOTE

I comandi possono essere concatenati con il punto e virgola (`;`). Abbiamo utilizzato il comando `echo` nell'esempio sopra riportato per stampare un carattere di nuova riga. Prova a eseguire semplicemente `cat /proc/1/cmdline` per vedere la differenza.

Il Carico di Sistema

In un sistema ogni processo può potenzialmente consumare risorse. Il così detto carico di sistema cerca di aggregare il carico complessivo del sistema in un unico indicatore numerico. Puoi vedere il carico corrente con il comando `uptime`:

```
$ uptime
 22:12:54 up 13 days, 20:26,  1 user,  load average: 2.91, 1.59, 0.39
```

Le ultime tre cifre indicano rispettivamente il carico medio del sistema nell'ultimo minuto (`2.91`), negli ultimi cinque minuti (`1.59`) e negli ultimi quindici minuti (`0.39`).

Ciascuno di questi numeri indica quanti processi erano in attesa o delle risorse della CPU o del completamento delle operazioni di input/output. Ciò significa che questi processi erano pronti per essere eseguiti se avessero ricevuto le rispettive risorse.

Log di Sistema e Messaggistica di Sistema

Non appena il kernel e i processi iniziano a essere eseguiti e a comunicare tra loro, vengono prodotte molte informazioni. La maggior parte viene inviata ai file: i cosiddetti file di log o, semplicemente, *log*.

Senza i log, la ricerca di un evento accaduto su un server darebbe molti grattacapi agli amministratori di sistema: da qui l'importanza di avere un modo standardizzato e centralizzato per tenere traccia di tutti gli eventi di sistema. Inoltre, i log sono fondamentali e danno indicazioni

quando si tratta di individuare e risolvere problemi e per la sicurezza, nonché sono fonti di dati affidabili per comprendere le statistiche di sistema e fare previsioni future.

Log con il Demone syslog

Tradizionalmente, i messaggi di sistema venivano gestiti dallo strumento di log standard, syslog, o da uno qualsiasi dei suoi derivati: syslog-ng o rsyslog. Il demone di log raccoglie i messaggi da altri servizi e programmi e li memorizza in file di log, tipicamente sotto `/var/log`. Tuttavia, alcuni servizi si occupano dei propri log (considera, per esempio, il server web Apache HTTPD). Allo stesso modo, il kernel Linux utilizza un ring buffer in memoria per archiviare i propri messaggi di log.

File di Log in `/var/log`

Poiché i log sono dati che variano nel tempo, normalmente si trovano in `/var/log`.

Se esplori `/var/log`, ti renderai conto che i nomi dei log sono, in una certo modo, abbastanza autoesplicativi. Alcuni esempi includono:

`/var/log/auth.log`
: Memorizza informazioni sull'autenticazione.

`/var/log/kern.log`
: Memorizza informazioni del kernel.

`/var/log/syslog`
: Memorizza informazioni di sistema.

`/var/log/messages`
: Memorizza i dati di sistema e delle applicazioni.

NOTE Il nome preciso e il contenuto dei file di log possono variare a seconda della distribuzione Linux utilizzata .

Accedere ai File di Log

Quando esplori i file di log, ricorda di essere *root* (se non hai i permessi di lettura) e usa un paginatore come `less`;

```
# less /var/log/messages
Jun  4 18:22:48 debian liblogging-stdlog:  [origin software="rsyslogd" swVersion="8.24.0" x-
```

```
pid="285" x-info="http://www.rsyslog.com"] rsyslogd was HUPed
Jun 29 16:57:10 debian kernel: [    0.000000] Linux version 4.9.0-8-amd64 (debian-
kernel@lists.debian.org) (gcc version 6.3.0 20170516 (Debian 6.3.0-18+deb9u1) ) #1 SMP
Debian 4.9.130-2 (2018-10-27)
Jun 29 16:57:10 debian kernel: [    0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-
8-amd64 root=/dev/sda1 ro quiet
```

In alternativa, puoi usare `tail` con l'opzione `-f` per leggere i messaggi più recenti del file e mostrare dinamicamente le nuove righe man mano che vengono aggiunte:

```
# tail -f /var/log/messages
Jul  9 18:39:37 debian kernel: [    2.350572] RAPL PMU: hw unit of domain psys 2^-0 Joules
Jul  9 18:39:37 debian kernel: [    2.512802] input: VirtualBox USB Tablet as
/devices/pci0000:00/0000:00:06.0/usb1/1-1/1-1:1.0/0003:80EE:0021.0001/input/input7
Jul  9 18:39:37 debian kernel: [    2.513861] Adding 1046524k swap on /dev/sda5.  Priority:-
1 extents:1 across:1046524k FS
Jul  9 18:39:37 debian kernel: [    2.519301] hid-generic 0003:80EE:0021.0001:
input,hidraw0: USB HID v1.10 Mouse [VirtualBox USB Tablet] on usb-0000:00:06.0-1/input0
Jul  9 18:39:37 debian kernel: [    2.623947] snd_intel8x0 0000:00:05.0: white list rate for
1028:0177 is 48000
Jul  9 18:39:37 debian kernel: [    2.914805] IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not
ready
Jul  9 18:39:39 debian kernel: [    4.937283] e1000: enp0s3 NIC Link is Up 1000 Mbps Full
Duplex, Flow Control: RX
Jul  9 18:39:39 debian kernel: [    4.938493] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link
becomes ready
Jul  9 18:39:40 debian kernel: [    5.315603] random: crng init done
Jul  9 18:39:40 debian kernel: [    5.315608] random: 7 urandom warning(s) missed due to
ratelimiting
```

L'output avrà il seguente formato:

- Timestamp
- Nome dell'host da cui proviene il messaggio
- Nome del programma/servizio che ha generato il messaggio
- Il PID del programma che ha generato il messaggio
- Descrizione del fatto che si è verificato

La maggior parte dei file di log sono scritti in testo normale; tuttavia, alcuni possono contenere dati binari come `/var/log/wtmp`, che memorizza i dati relativi agli accessi riusciti. Puoi utilizzare

il comando `file` per determinare di volta in volta qual è il caso:

```
$ file /var/log/wtmp
/var/log/wtmp: dBase III DBT, version number 0, next free block index 8
```

Questi file vengono normalmente letti utilizzando comandi speciali. `last` viene utilizzato per interpretare i dati in `/var/log/wtmp`:

```
$ last
carol    tty2         :0               Thu May 30 10:53   still logged in
reboot   system boot  4.9.0-9-amd64    Thu May 30 10:52   still running
carol    tty2         :0               Thu May 30 10:47 - crash  (00:05)
reboot   system boot  4.9.0-9-amd64    Thu May 30 09:11   still running
carol    tty2         :0               Tue May 28 08:28 - 14:11  (05:42)
reboot   system boot  4.9.0-9-amd64    Tue May 28 08:27 - 14:11  (05:43)
carol    tty2         :0               Mon May 27 19:40 - 19:52  (00:11)
reboot   system boot  4.9.0-9-amd64    Mon May 27 19:38 - 19:52  (00:13)
carol    tty2         :0               Mon May 27 19:35 - down   (00:03)
reboot   system boot  4.9.0-9-amd64    Mon May 27 19:34 - 19:38  (00:04)
```

NOTE Analogamente a `/var/log/wtmp`, `/ var/log/btmp` memorizza le informazioni sui tentativi di accesso falliti e il comando speciale per leggerne il contenuto è `lastb`.

Rotazione dei Log

I file di log possono crescere molto in poche settimane o in pochi mesi e occupare tutto lo spazio libero sul disco. Per risolvere questo problema, viene utilizzata l'utility `logrotate`. Implementa la rotazione o il ciclo dei log che implica azioni come rinominare i file di log, archiviarli e/o comprimerli, a volte inviandoli tramite e-mail all'amministratore di sistema, ed eventualmente eliminandoli man mano che invecchiano. Esistono diverse convenzioni utilizzate per assegnare i nomi a questi file di log ruotati (per esempio l'aggiunta di un suffisso con la data); tuttavia, è pratica comune aggiungere semplicemente un suffisso con un numero intero:

```
# ls /var/log/apache2/
access.log  error.log  error.log.1  error.log.2.gz  other_vhosts_access.log
```

Nota come `error.log.2.gz` sia già stato compresso con `gzip` (da qui il suffisso `.gz`).

Il Kernel Ring Buffer

Il kernel ring buffer è una struttura dati a dimensione fissa che registra i messaggi del kernel sia durante il processo di avvio sia in tempo reale. La funzione di questo buffer è molto importante e consiste nel registrare tutti i messaggi del kernel generati all'avvio quando `syslog` non è ancora disponibile. Il comando `dmesg` stampa il kernel ring buffer (che era anche memorizzato in `/var/log/dmesg`). A causa delle grandi dimensioni del ring buffer, questo comando viene normalmente utilizzato in combinazione con l'utilità di filtraggio del testo `grep` o con un paginatore come `less`. Per esempio, per cercare i messaggi di avvio:

```
$ dmesg | grep boot
[    0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64 root=UUID=5216e1e4-ae0e-
441f-b8f5-8061c0034c74 ro quiet
[    0.000000] smpboot: Allowing 1 CPUs, 0 hotplug CPUs
[    0.000000] Kernel command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64
root=UUID=5216e1e4-ae0e-441f-b8f5-8061c0034c74 ro quiet
[    0.144986] AppArmor: AppArmor disabled by boot time parameter
(...)
```

NOTE Man mano che il kernel ring buffer cresce nel tempo con nuovi messaggi, quelli più vecchi scompaiono.

Il Journal di Sistema: systemd-journald

A partire dal 2015, systemd ha sostituito SysV Init come gestore di sistema e dei servizi *de facto* nella maggior parte delle principali distribuzioni Linux. Di conseguenza, il demone del journal — journald — è diventato il componente di log standard, sostituendo syslog per la maggior parte degli aspetti. I dati non vengono più memorizzati in testo normale ma in formato binario. Quindi, l'utility `journalctl` è necessaria per leggere i log. Inoltre, journald è compatibile con syslog e può essere integrato con esso.

`journalctl` è l'utility usata per leggere e interrogare il database del journal di systemd. Se invocato senza opzioni, stampa l'intero journal:

```
# journalctl
-- Logs begin at Tue 2019-06-04 17:49:40 CEST, end at Tue 2019-06-04 18:13:10 CEST. --
jun 04 17:49:40 debian systemd-journald[339]: Runtime journal (/run/log/journal/) is 8.0M,
max 159.6M, 151.6M free.
jun 04 17:49:40 debian kernel: microcode: microcode updated early to revision 0xcc, date =
2019-04-01
Jun 04 17:49:40 debian kernel: Linux version 4.9.0-8-amd64 (debian-kernel@lists.debian.org)
```

```
(gcc version 6.3.0 20170516 (Debian 6.3.0-18+deb9u1) )
Jun 04 17:49:40 debian kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-8-amd64
root=/dev/sda1 ro quiet
(...)
```

Tuttavia, se invocato con le opzioni `-k` o `--dmesg`, è equivalente all'uso del comando `dmesg`:

```
# journalctl -k
[    0.000000] Linux version 4.9.0-9-amd64 (debian-kernel@lists.debian.org) (gcc version
6.3.0 20170516 (Debian 6.3.0-18+deb9u1) ) #1 SMP Debian 4.9.168-1+deb9u2 (2019-05-13)
[    0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64 root=UUID=5216e1e4-ae0e-
441f-b8f5-8061c0034c74 ro quiet
(...)
```

Altre opzioni interessanti per `journalctl` sono:

`-b`, `--boot`

Mostra le informazioni di avvio.

`-u`

Mostra i messaggi relativi a un'unità specificata. Grosso modo, una unità può essere definita come una qualsiasi risorsa gestita da *systemd*. Per esempio, il comando `journalctl -u apache2.service` viene utilizzato per leggere i messaggi relativi al server web `apache2`.

`-f`

Mostra i messaggi più recenti del journal e continua a stampare le nuove voci man mano che vengono aggiunte al journal, in modo molto simile a `tail -f`.

Esercizi Guidati

1. Dai un'occhiata al seguente elenco prodotto dal comando `top` e rispondi alle domande di seguito riportate:

```
carol@debian:~$ top

top - 13:39:16 up 31 min,  1 user,  load average: 0.12, 0.15, 0.10
Tasks:  73 total,   2 running,  71 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.1 us,  0.4 sy,  0.0 ni, 98.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  1020332 total,   698700 free,   170664 used,   150968 buff/cache
KiB Swap:  1046524 total,  1046524 free,        0 used.   710956 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
  605 nobody    20   0 1137620 132424  34256 S  6.3 13.0   1:47.24 ntopng
  444 www-data  20   0  364780   4132   2572 S  0.3  0.4   0:00.44 apache2
  734 root      20   0   95212   7004   6036 S  0.3  0.7   0:00.36 sshd
  887 carol     20   0   46608   3680   3104 R  0.3  0.4   0:00.03 top
    1 root      20   0   56988   6688   5240 S  0.0  0.7   0:00.42 systemd
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S  0.0  0.0   0:00.09 ksoftirqd/0
    4 root      20   0       0      0      0 S  0.0  0.0   0:00.87 kworker/0:0
(...)
```

   - Quali processi sono stati avviati dall'utente `carol`?

   - Quale directory virtuale in `/proc` dovresti visitare per cercare i dati riguardanti il comando `top`?

   - Quale processo è stato eseguito per primo? Come l'hai capito?

   - Completa la seguente tabella specificando in quale zona dell'output di `top` si trovano le seguenti informazioni:

Informazioni su ...	**Area di Riepilogo**	**Area delle Attività**
Memoria		
Swap		

Informazioni su ...	**Area di Riepilogo**	**Area delle Attività**
PID		
Tempo di CPU		
Comandi		

2. Quale comando viene utilizzato per leggere i seguenti log binari?
   - `/var/log/wtmp`
   - `/var/log/btmp`
   - `/run/log/journal/2a7d9730cd3142f4b15e20d6be631836/system.journal`
3. Quali comandi potresti utilizzare, in combinazione con `grep`, per scoprire le seguenti informazioni sul tuo sistema Linux?
   - Quando il sistema è stato riavviato l'ultima volta (`wtmp`)
   - Quali dischi rigidi sono installati (`kern.log`)
   - Quando è avvenuto l'ultimo login (`auth.log`)
4. Indica i due comandi che potresti utilizzare per visualizzare il kernel ring buffer.
5. Indica dove puoi trovare i seguenti messaggi di log:
   - `Jul 10 13:37:39 debian dbus[303]: [system] Successfully activated service 'org.freedesktop.nm_dispatcher'`

`/var/log/auth.log`	
`/var/log/kern.log`	
`/var/log/syslog`	
`/var/log/messages`	

- `Jul 10 11:23:58 debian kernel: [ 1.923349] usbhid: USB HID core driver`

`/var/log/auth.log`	
`/var/log/kern.log`	
`/var/log/syslog`	
`/var/log/messages`	

- `Jul 10 14:02:53 debian sudo: pam_unix(sudo:session): session opened for user root by carol(uid=0)`

`/var/log/auth.log`	
`/var/log/kern.log`	
`/var/log/syslog`	
`/var/log/messages`	

- `Jul 10 11:23:58 debian NetworkManager[322]: <info> [1562750638.8672] NetworkManager (version 1.6.2) is starting…`

`/var/log/auth.log`	
`/var/log/kern.log`	
`/var/log/syslog`	
`/var/log/messages`	

6. Usa `journalctl` per trovare informazioni sulle seguenti unità:

Unità	**Comando**
`ssh`	
`networking`	
`rsyslog`	
`cron`	

Esercizi Esplorativi

1. Prendi nuovamente in considerazione l'output del comando `top` mostrato negli esercizi guidati e rispondi alle seguenti domande:
   - Quali sono i due passaggi per terminare il server web *apache*?
   - Nell'area di riepilogo, come puoi visualizzare le informazioni sulla memoria fisica e di swap utilizzando delle barre di avanzamento?
   - Ora, ordina i processi in base all'utilizzo della memoria:
   - Ora che le informazioni sulla memoria sono visualizzate con delle barre di avanzamento e che i processi sono ordinati in base all'utilizzo della memoria, salva queste configurazioni in modo da averle come predefinite per la prossima volta che utilizzerai `top`:
   - Quale file memorizza le impostazioni di configurazione di `top`? Dove si trova? Come puoi verificarne l'esistenza?
2. Approfondisci il comando `exec` in Bash. Prova a mostrarne il funzionamento: avvia una sessione Bash e trova il processo Bash con `ps`; esegui poi `exec /bin/sh` e cerca di nuovo il processo con lo stesso PID.
3. Segui questi passaggi per esplorare gli eventi del kernel e la gestione dinamica dei dispositivi da parte di udev:
   - Collega un'unità USB al tuo computer. Esegui `dmesg` e presta attenzione alle ultime righe. Qual è la riga più recente?
   - Tenendo presente l'output del comando precedente, esegui `ls /dev/sd*` e assicurati che la tua unità USB sia visualizzata nell'elenco. Qual è l'output?
   - Ora rimuovi l'unità USB ed esegui di nuovo `dmesg`. Che cosa mostra la riga più recente?

- Esegui di nuovo `ls /dev/sd*` e assicurati che il tuo dispositivo sia scomparso dall'elenco. Qual è l'output?

Sommario

Nel contesto dell'archiviazione dei dati, in questa lezione sono stati discussi i seguenti argomenti: gestione dei processi e log e messaggistica di sistema.

Per quanto riguarda la gestione dei processi, abbiamo appreso quanto segue:

- I programmi generano processi e i processi sono organizzati gerarchicamente;
- Ogni processo ha un identificativo univoco (`PID`) e un identificativo del processo padre (`PPID`);
- `top` è un comando molto utile per esplorare in modo dinamico e interattivo i processi in esecuzione in un sistema;
- È possibile utilizzare `ps` per ottenere un'istantanea dei processi correnti in esecuzione in un sistema;
- La directory `/proc` include delle directory per ogni processo in esecuzione nel sistema che prendono il nome dai relativi PID;
- Il concetto di carico medio del sistema, che è molto utile per controllare l'utilizzo/sovraccarico della CPU.

Per quanto riguarda i log di sistema, dobbiamo ricordare che:

- Un log è un file in cui vengono registrati gli eventi di sistema. I log sono preziosi per la risoluzione dei problemi;
- I log sono stati tradizionalmente gestiti da servizi speciali come syslog, syslog-ng o rsyslog. Tuttavia, alcuni programmi utilizzano i propri demoni di log;
- Poiché i log sono dati variabili, sono conservati in `/var` e, a volte, i loro nomi possono darti un'idea del loro contenuto (`kern.log`, `auth.log`, etc.);
- La maggior parte dei log sono scritti in testo normale e possono essere letti con un qualsiasi editor di testo purché si disponga delle autorizzazioni appropriate. Tuttavia, alcuni di essi sono binari e devono essere letti utilizzando comandi speciali;
- Per evitare problemi di spazio su disco, l'utility logrotate esegue la *rotazione dei log*;
- Per quanto riguarda il kernel, utilizza una struttura dati circolare, il ring buffer, dove vengono conservati i messaggi di avvio (i vecchi messaggi scompaiono nel tempo);
- Il gestore di sistema e dei servizi systemd ha sostituito *System V init* praticamente in tutte le distribuzioni e journald è diventato il servizio di log standard;
- Per leggere il journal di systemd, è necessaria l'utility `journalctl`.

Comandi utilizzati in questa lezione:

`cat`

Concatena/stampa il contenuto di un file.

`dmesg`

Stampa il kernel ring buffer.

`echo`

Visualizza una riga di testo o una nuova riga.

`file`

Determina il tipo di file.

`grep`

Stampa le righe che hanno corrispondenza con un pattern.

`last`

Mostra un elenco degli ultimi utenti che hanno effettuato l'accesso.

`less`

Visualizza il contenuto di un file una pagina alla volta.

`ls`

Elenca il contenuto di una directory.

`journalctl`

Interroga il journal di `systemd`.

`tail`

Visualizza le ultime righe di un file.

Risposte agli Esercizi Guidati

1. Dai un'occhiata al seguente elenco prodotto dal comando `top` e rispondi alle domande di seguito riportate:

```
carol@debian:~$ top

top - 13:39:16 up 31 min,  1 user,  load average: 0.12, 0.15, 0.10
Tasks:  73 total,   2 running,  71 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.1 us,  0.4 sy,  0.0 ni, 98.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  1020332 total,   698700 free,   170664 used,   150968 buff/cache
KiB Swap:  1046524 total,  1046524 free,        0 used.   710956 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
  605 nobody    20   0 1137620 132424  34256 S  6.3 13.0   1:47.24 ntopng
  444 www-data  20   0  364780   4132   2572 S  0.3  0.4   0:00.44 apache2
  734 root      20   0   95212   7004   6036 S  0.3  0.7   0:00.36 sshd
  887 carol     20   0   46608   3680   3104 R  0.3  0.4   0:00.03 top
    1 root      20   0   56988   6688   5240 S  0.0  0.7   0:00.42 systemd
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S  0.0  0.0   0:00.09 ksoftirqd/0
    4 root      20   0       0      0      0 S  0.0  0.0   0:00.87 kworker/0:0
(...)
```

   - Quali processi sono stati avviati dall'utente `carol`?

     Soluzione: Solo uno: `top`.

   - Quale directory virtuale in `/proc` dovresti visitare per cercare i dati riguardanti il comando `top`?

     Soluzione: `/proc/887`

   - Quale processo è stato eseguito per primo? Come l'hai capito?

     Soluzione: `systemd`, poiché è quello con `PID` #1.

   - Completa la seguente tabella specificando in quale zona dell'output di `top` si trovano le seguenti informazioni:

Informazioni su ...	**Area di Riepilogo**	**Area delle Attività**
Memoria	Si	Si

Informazioni su ...	Area di Riepilogo	Area delle Attività
Swap	Si	No
`PID`	No	Si
Tempo di CPU	Si	Si
Comandi	No	Si

2. Quale comando viene utilizzato per leggere i seguenti log binari?
   - `/var/log/wtmp`

     Soluzione: `last`
   - `/var/log/btmp`

     Soluzione: `lastb`
   - `/run/log/journal/2a7d9730cd3142f4b15e20d6be631836/system.journal`

     Soluzione: `journalctl`
3. Quali comandi potresti utilizzare, in combinazione con `grep`, per scoprire le seguenti informazioni sul tuo sistema Linux?
   - Quando il sistema è stato riavviato l'ultima volta (`wtmp`)

     Soluzione: `last`
   - Quali dischi rigidi sono installati (`kern.log`)

     Soluzione: `less /var/log/kern.log`
   - Quando è avvenuto l'ultimo login (`auth.log`)

     Soluzione: `less /var/log/auth.log`
4. Indica i due comandi che potresti utilizzare per visualizzare il kernel ring buffer.

   `dmesg` e `journalctl -k` (o anche `journalctl --dmesg`).
5. Indica dove puoi trovare i seguenti messaggi di log:
   - `Jul 10 13:37:39 debian dbus[303]: [system] Successfully activated service 'org.freedesktop.nm_dispatcher'`

`/var/log/auth.log`	
`/var/log/kern.log`	
`/var/log/syslog`	X
`/var/log/messages`	

- `Jul 10 11:23:58 debian kernel: [ 1.923349] usbhid: USB HID core driver`

`/var/log/auth.log`	
`/var/log/kern.log`	X
`/var/log/syslog`	
`/var/log/messages`	X

`Jul 10 14:02:53 debian sudo: pam_unix(sudo:session): session opened for user root by carol(uid=0)`

`/var/log/auth.log`	X
`/var/log/kern.log`	
`/var/log/syslog`	
`/var/log/messages`	

- `Jul 10 11:23:58 debian NetworkManager[322]: <info> [1562750638.8672] NetworkManager (version 1.6.2) is starting…`

`/var/log/auth.log`	
`/var/log/kern.log`	
`/var/log/syslog`	
`/var/log/messages`	X

6. Usa `journalctl` per trovare informazioni sulle seguenti unità:

Unità	**Comando**
`ssh`	`journalctl -u ssh.service`
`networking`	`journalctl -u networking.service`
`rsyslog`	`journalctl -u rsyslog.service`

Unità	Comando
`cron`	`journalctl -u cron.service`

Risposte agli Esercizi Esplorativi

1. Prendi nuovamente in considerazione l'output del comando `top` mostrato negli esercizi guidati e rispondi alle seguenti domande:
   - Quali sono i due passaggi per terminare il server web *apache*?

     Prima premi `k` e poi indica un valore per `kill`.

   - Nell'area di riepilogo, come puoi visualizzare le informazioni sulla memoria fisica e di swap utilizzando delle barre di avanzamento?

     Premendo `m` una o due volte.

   - Ora, ordina i processi in base all'utilizzo della memoria:

     `M`

   - Ora che le informazioni sulla memoria sono visualizzate con delle barre di avanzamento e che i processi sono ordinati in base all'utilizzo della memoria, salva queste configurazioni in modo da averle come predefinite per la prossima volta che utilizzerai `top`:

     `W`

   - Quale file memorizza le impostazioni di configurazione di `top`? Dove si trova? Come puoi verificarne l'esistenza?

     Il file è `~/.config/procps/toprc` e si trova nella directory home dell'utente (`~`). Dato che è un file nascosto (si trova in una directory il cui nome inizia con un punto), possiamo verificarne l'esistenza con `ls -a` (il comando elenca tutti i file). Questo file può essere generato premendo `Shift` + `W` mentre ci si trova in `top`.

2. Approfondisci il comando `exec` in Bash. Prova a mostrarne il funzionamento: avvia una sessione Bash e trova il processo Bash con `ps`; esegui poi `exec /bin/sh` e cerca di nuovo il processo con lo stesso PID.

   `exec` sostituisce un processo con un altro comando. Nell'esempio di seguito riportato possiamo vedere che il processo Bash viene sostituito da `/bin/sh` (invece che dal diventare `/bin/sh` un processo figlio):

```
$ echo $$
19877
$ ps auxf | grep 19877 | head -1
```

```
carol  19877  0.0  0.0   7448  3984 pts/25   Ss   21:17   0:00  \_ bash
$ exec /bin/sh
sh-5.0$ ps auxf | grep 19877 | head -1
carol  19877  0.0  0.0   7448  3896 pts/25   Ss   21:17   0:00  \_ /bin/sh
```

3. Segui questi passaggi per esplorare gli eventi del kernel e la gestione dinamica dei dispositivi da parte di udev:
   - Collega un'unità USB al tuo computer. Esegui `dmesg` e presta attenzione alle ultime righe. Qual è la riga più recente?

     Dovresti ottenere qualcosa di simile a `[ 1967.700468] sd 6:0:0:0: [sdb] Attached SCSI removable disk`.

   - Tenendo presente l'output del comando precedente, esegui `ls /dev/sd*` e assicurati che la tua unità USB appaia nell'elenco. Qual è l'output?

     A seconda del numero di dispositivi collegati al tuo sistema, dovresti ottenere qualcosa di simile a `/dev/sda /dev/sda1 /dev/sdb /dev/sdb1 /dev/sdb2`. Nel nostro caso troviamo la nostra unità USB (`/dev/sdb`) e le sue due partizioni (`/dev/sdb1` e `/dev/sdb2`).

   - Ora rimuovi l'unità USB ed esegui di nuovo `dmesg`. Che cosa mostra la riga più recente?

     Dovresti ottenere qualcosa di simile a `[ 2458.881695] usb 1-9: USB disconnect, device number 6`.

   - Esegui di nuovo `ls /dev/sd*` e assicurati che il tuo dispositivo sia scomparso dall'elenco. Qual è l'output?

     Nel nostro caso, `/dev/sda /dev/sda1`.

4.4 Il Tuo Computer in Rete

Obiettivi LPI di riferimento

Linux Essentials version 1.6, Exam 010, Objective 4.4

Peso

2

Aree di Conoscenza Chiave

- Internet, rete, router
- Esaminare la configurazione del client DNS
- Esaminare la configurazione di rete

Di seguito è riportato un elenco parziale dei file, dei termini e dei comandi utilizzati

- `route`, `ip route show`
- `ifconfig`, `ip addr show`
- `netstat`, `ss`
- `/etc/resolv.conf`, `/etc/hosts`
- `IPv4`, `IPv6`
- `ping`
- `host`

4.4 Lezione 1

Certificazione:	Linux Essentials
Versione:	1.6
Argomento:	4 Il Sistema Operativo Linux
Obiettivo:	4.4 Il Tuo Computer in Rete
Lezione:	1 di 1

Introduzione

Al giorno d'oggi, qualsiasi tipo di dispositivo informatico scambia informazioni sulle reti. L'idea alla base delle reti di computer è la connessione fisica tra un dispositivo e i suoi *peer*. Queste connessioni sono chiamate *link* e sono le connessioni più semplici tra due diversi dispositivi. I link possono essere stabiliti attraverso vari mezzi, come cavi in rame, fibre ottiche, onde radio o laser.

Ogni link è collegato a un'interfaccia di un dispositivo. Ogni dispositivo può avere più interfacce e quindi essere connesso a più link. Attraverso questi link i computer possono formare una rete: una piccola comunità di dispositivi che possono connettersi direttamente tra loro. Esistono numerosi esempi di reti di questo tipo nel mondo. Per essere in grado di comunicare al di fuori di una rete *link layer* (a livello di link), i dispositivi utilizzano i router. Pensa alle reti link layer come isole collegate da router, proprio come da ponti, sui quali viaggiano le informazioni per raggiungere un dispositivo che fa parte di un'isola remota.

Questo modello porta a diversi livelli di rete:

Link Layer

Gestisce la comunicazione tra dispositivi collegati direttamente.

 | Questa opera è concessa in licenza CC BY-NC-ND 4.0. |

Network Layer

Gestisce il *routing* al di fuori delle singole reti e l'indirizzamento univoco dei dispositivi al di fuori di una rete link layer.

Application Layer

Consente ai singoli programmi di connettersi tra loro.

Quando furono inventate, le reti di computer usavano gli stessi metodi di comunicazione dei telefoni in quanto erano a commutazione di circuito. Ciò significa che era necessario creare un link dedicato e diretto tra due nodi affinché fossero in grado di comunicare. Questo metodo funzionava bene, ma richiedeva tutto lo spazio su un dato link perché solo due host potessero comunicare.

Successivamente le reti di computer sono passate a un meccanismo chiamato *commutazione di pacchetto*. Secondo questo metodo i dati vengono raggruppati con un *header* (intestazione), che contiene informazioni sull'origine e sulla destinazione. Le informazioni sul contenuto effettivo sono contenute in questo *frame* e inviate tramite il link al destinatario indicato nell'header del frame. Ciò consente a più dispositivi di condividere un singolo link e comunicare quasi contemporaneamente.

Rete Link Layer

Il compito di ogni pacchetto è trasportare le informazioni dalla sorgente alla destinazione attraverso un link che collega entrambi i dispositivi. Questi dispositivi hanno bisogno di un modo per identificarsi reciprocamente. Questo è lo scopo di un *indirizzo di link layer* (indirizzo a livello di link). In una rete Ethernet, gli *indirizzi MAC* (Media Access Control) vengono utilizzati per identificare i singoli dispositivi. Un indirizzo MAC è composto da 48 bit. Non sono necessariamente univoci a livello globale e non possono essere utilizzati per indirizzare peer al di fuori del collegamento corrente. Pertanto, questi indirizzi non possono essere utilizzati per instradare i pacchetti verso altri collegamenti. Il destinatario di un pacchetto controlla se l'indirizzo di destinazione corrisponde al proprio indirizzo di link layer e, in caso affermativo, elabora il pacchetto. Altrimenti il pacchetto viene scartato. L'eccezione a questa regola è rappresentata dai *pacchetti di broadcast* (un pacchetto inviato a chiunque in una rete locale) che sono sempre accettati.

Il comando `ip link show` mostra un elenco di tutte le interfacce di rete disponibili e dei loro indirizzi di link layer, nonché alcune altre informazioni sulla dimensione massima del pacchetto:

```
$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group
```

```
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT
group default qlen 1000
    link/ether 00:0c:29:33:3b:25 brd ff:ff:ff:ff:ff:ff
```

L'output sopra riportato mostra che il dispositivo ha due interfacce, `lo` e `ens33`. `lo` è il *dispositivo di loopback* e ha indirizzo MAC `00:00:00:00:00:00`, mentre `ens33` è un'interfaccia ethernet e ha indirizzo MAC `00:0c:29:33:3b:25`.

Rete IPv4

Per visitare siti web come Google o Twitter, per controllare la posta elettronica o per consentire alle aziende di connettersi tra loro, i pacchetti devono essere in grado di spostarsi da una rete link layer all'altra. Spesso queste reti sono collegate solo indirettamente, con diverse reti link layer intermedie che i pacchetti devono attraversare per raggiungere l'effettiva destinazione.

Gli indirizzi di link layer di un'interfaccia di rete non possono essere utilizzati al di fuori di quella specifica rete link layer. Poiché questo indirizzo non ha alcun significato per i dispositivi in altre reti link layer, è necessaria una forma di indirizzi univoci a livello globale per implementare il routing. Questo schema di indirizzamento, insieme al concetto generale di routing, è implementato dall'*Internet Protocol* (IP).

NOTE

Un *protocollo* è un insieme di procedure per fare qualcosa in modo che tutte le parti che seguono il protocollo siano compatibili tra loro. Un protocollo può essere visto come la definizione di uno standard. In informatica, l'Internet Protocol è uno standard accettato da tutti in modo che diversi dispositivi prodotti da diversi produttori possano comunicare tra loro.

Indirizzi IPv4

Gli indirizzi IP, come gli indirizzi MAC, rappresentano un modo per indicare da dove proviene un pacchetto dati e dove sta andando. IPv4 è stato il primo protocollo. Gli indirizzi IPv4 hanno 32 bit, fornendo un numero massimo teorico di 4.294.967.296 indirizzi. Tuttavia, il numero di questi indirizzi utilizzabili dai dispositivi è molto inferiore in quanto alcuni intervalli sono riservati per usi speciali come, per esempio, gli indirizzi broadcast (che vengono utilizzati per raggiungere tutti i partecipanti di una specifica rete), gli indirizzi multicast (simili agli indirizzi broadcast, ma il dispositivo deve sintonizzarsi come una radio) o quelli riservati per uso privato.

Gli indirizzi IPv4 sono rappresentati da quattro cifre separate da un punto e ogni cifra può variare da `0` a `255`. Per esempio, considera il seguente indirizzo IP:

```
192.168.0.1
```

Tecnicamente, ciascuna di queste cifre rappresenta otto singoli bit. Quindi questo indirizzo può anche essere scritto in questo modo:

```
11000000.10101000.00000000.00000001
```

All'atto pratico viene utilizzata la notazione decimale come visto sopra. Tuttavia, la rappresentazione bit per bit è importante per comprendere il *subnetting*.

Sottoreti IPv4

Per supportare il routing, gli indirizzi IP possono essere suddivisi in due parti: la porzione di rete e la porzione host. La porzione di rete identifica la rete su cui si trova il dispositivo e viene utilizzata per instradare i pacchetti verso quella rete. La porzione *host* viene utilizzata per identificare in modo specifico un dato dispositivo su una rete e per consegnare il pacchetto allo specifico destinatario una volta che ha raggiunto la sua rete link layer.

Gli indirizzi IP possono essere suddivisi nella porzione di sottorete e nella porzione host in qualsiasi punto. La cosiddetta *maschera di sottorete* (*subnet mask*), chiamata anche *maschera di rete* (*netmask*), definisce dove avviene questa divisione. Prendiamo nuovamente in considerazione la rappresentazione binaria dell'indirizzo IP del precedente esempio:

```
11000000.10101000.00000000.00000001
```

Per questo indirizzo IP, la subnet mask imposta ogni bit che appartiene alla porzione di rete a `1` e ogni bit che appartiene alla porzione host a `0`:

```
11111111.11111111.11111111.00000000
```

All'atto pratico la netmask viene scritta in notazione decimale:

```
255.255.255.0
```

Ciò significa che questa rete va da `192.168.0.0` a `192.168.0.255`. Nota che i primi tre numeri, per i quali sono impostati tutti i bit nella maschera di rete, rimangono invariati negli indirizzi IP.

Infine, c'è una notazione alternativa per la subnet mask, che viene chiamata *Classless Inter-*

Domain Routing (CIDR). Questa notazione indica semplicemente quanti bit sono impostati nella subnet mask e aggiunge tale numero all'indirizzo IP. Nell'esempio precedente, 24 bit su 32 sono impostati a `1` nella subnet mask. Questo può essere espresso secondo la notazione CIDR come `192.168.0.1/24`.

Indirizzi Privati IPv4

Come accennato in precedenza, alcune sezioni dello spazio degli indirizzi IPv4 sono riservate per usi speciali. Uno di questi è l'assegnazione di indirizzi privati. Le seguenti sottoreti sono riservate per l'indirizzamento privato:

- `10.0.0.0/8`
- `172.16.0.0/12`
- `192.168.0.0/16`

Chiunque può utilizzare gli indirizzi di queste sottoreti. Tuttavia, queste sottoreti non possono essere instradate sulla rete Internet pubblica poiché sono potenzialmente utilizzate da numerose reti in contemporanea.

Oggi la maggior parte delle reti utilizza questi indirizzi interni. Consentono la comunicazione interna senza la necessità di alcuna assegnazione di indirizzi esterni. La maggior parte delle connessioni Internet oggi viene fornita con un unico indirizzo IPv4 esterno. Quando inoltrano i pacchetti a Internet, i router mappano tutti gli indirizzi interni su quell'unico indirizzo IP esterno. Questo è chiamato *Network Address Translation* (NAT). Il caso speciale di NAT in cui un router mappa gli indirizzi interni su un singolo indirizzo IP esterno è talvolta chiamato *mascheramento* (*masquerading*). Ciò consente a qualsiasi dispositivo sulla rete interna di stabilire nuove connessioni con qualsiasi indirizzo IP globale in Internet.

NOTE Con il masquerading non è possibile fare riferimento ai dispositivi interni da Internet poiché non hanno un indirizzo valido a livello globale. Tuttavia, questa non rappresenta una caratteristica di sicurezza. Anche quando si utilizza il masquerading, è comunque necessario un firewall.

Configurazione di Indirizzi IPv4

Esistono due modi principali per configurare gli indirizzi IPv4 su un computer: o assegnando gli indirizzi manualmente o utilizzando il *Dynamic Host Configuration Protocol* (DHCP) per la configurazione automatica.

Quando si utilizza il DHCP, un server centrale controlla quali indirizzi vengono distribuiti a quali dispositivi. Il server può anche fornire ai dispositivi altre informazioni sulla rete come, per

esempio, gli indirizzi IP dei server DNS, l'indirizzo IP del *default router*, o, nel caso di configurazioni più complesse, può avviare un sistema operativo dalla rete. Il DHCP è abilitato per impostazione predefinita su molti sistemi; quindi probabilmente avrai già un indirizzo IP una volta connesso a una rete.

Gli indirizzi IP possono anche essere aggiunti manualmente a un'interfaccia utilizzando il comando `ip addr add`. Nel seguente esempio aggiungiamo l'indirizzo `192.168.0.5` all'interfaccia `ens33`. La rete utilizza la netmask `255.255.255.0` che equivale a `/24` secondo la notazione CIDR:

```
$ sudo ip addr add 192.168.0.5/255.255.255.0 dev ens33
```

Ora possiamo verificare che l'indirizzo sia stato aggiunto utilizzando il comando `ip addr show`:

```
$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
25: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen
1000
    link/ether 00:0c:29:33:3b:25 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.5/24 192.168.0.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::010c:29ff:fe33:3b25/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

L'output mostra sia l'interfaccia `lo` sia l'interfaccia `ens33` con il relativo indirizzo, assegnato tramite il precedente comando.

Per verificare che un dispositivo sia raggiungibile è possibile utilizzare il comando `ping`. Questo invia un tipo speciale di messaggio chiamato *echo request* in cui il mittente chiede una risposta al destinatario. Se la connessione tra i due dispositivi può essere stabilita correttamente, il destinatario invierà un *echo reply*, verificando così la connessione:

```
$ ping -c 3 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=64 time=2.16 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=64 time=1.85 ms
```

```
64 bytes from 192.168.0.1: icmp_seq=3 ttl=64 time=3.41 ms

--- 192.168.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 5ms
rtt min/avg/max/mdev = 1.849/2.473/3.410/0.674 ms
```

L'opzione `-c 3` interrompe il comando `ping` dopo aver inviato tre *echo request*. Altrimenti, `ping` continua a funzionare all'infinito e deve essere fermato premendo `Ctrl` + `C`.

Routing IPv4

Il routing è il processo tramite il quale un pacchetto passa dalla rete di origine alla rete di destinazione. Ogni dispositivo mantiene una tabella di routing che contiene informazioni sulle reti IP che possono essere raggiunte direttamente tramite il collegamento del dispositivo alle reti link layer, e informazioni sulle reti IP che possono essere raggiunte inoltrando i pacchetti a un router. Infine, una *default route* definisce un router che riceve tutti i pacchetti che non hanno corrispondenza con alcun altro percorso.

Quando si stabilisce una connessione, il dispositivo cerca l'indirizzo IP della destinazione nella sua tabella di routing. Se una voce ha corrispondenza con l'indirizzo, il pacchetto viene inviato o alla rispettiva rete link layer o inoltrato al router indicato nella tabella di routing.

Anche i router stessi mantengono delle tabelle di routing. Quando riceve un pacchetto, un router cerca l'indirizzo di destinazione nella propria tabella di routing e invia il pacchetto al router successivo. Questo viene ripetuto fino a quando il pacchetto arriva al router sulla rete di destinazione. Ogni router coinvolto in questo percorso è chiamato *hop*. Quest'ultimo router individua, nella sua tabella di routing, un link diretto per l'indirizzo di destinazione e invia i pacchetti alla sua interfaccia.

La maggior parte delle reti domestiche ha solo un punto di uscita: il singolo router fornito dall'*internet service provider* (ISP). In questo caso, un dispositivo semplicemente inoltra tutti i pacchetti che non sono per la rete interna direttamente al router di casa che li invia quindi al router del provider per un ulteriore inoltro. Questo è un esempio di default route.

Il comando `ip route show` elenca la tabella di routing IPv4 corrente:

```
$ ip route show
127.0.0.0/8 via 127.0.0.1 dev lo0
192.168.0.0/24 dev ens33 scope link
```

Per aggiungere una default route, tutto ciò che serve è l'indirizzo interno del router che sarà il

gateway predefinito. Se, per esempio, il router ha l'indirizzo `192.168.0.1`, possiamo usare il seguente comando per configurarlo come default route:.

```
$ sudo ip route add default via 192.168.0.1
```

Per fare una verifica, esegui nuovamente `ip route show`:

```
$ ip route show
default via 192.168.0.1 dev ens33
127.0.0.0/8 via 127.0.0.1 dev lo0
192.168.0.0/24 dev ens33 scope link
```

Rete IPv6

IPv6 è stato progettato per risolvere le lacune di IPv4, in particolare la mancanza di indirizzi poiché sempre più dispositivi venivano messi online. Tuttavia, IPv6 include anche altre funzionalità come nuovi protocolli per la configurazione automatica della rete. Invece di 32 bit per indirizzo, IPv6 ne utilizza 128. Ciò consente circa 2^{128} indirizzi. Tuttavia, come IPv4, il numero di indirizzi utilizzabili univoci a livello globale è molto inferiore poichè alcune sezioni sono riservate per altri usi. Questo gran numero di indirizzi significa che ci sono indirizzi pubblici più che sufficienti per ogni dispositivo attualmente connesso a Internet e per molti altri che ci saranno in futuro, riducendo così la necessità di mascheramento e i problemi da esso generati come il ritardo nella traduzione e l'impossibilità di connettersi direttamente a dispositivi mascherati.

Indirizzi IPv6

Gli indirizzi utilizzano 8 gruppi di 4 cifre esadecimali ciascuno separati da due punti:

```
2001:0db8:0000:abcd:0000:0000:0000:7334
```

NOTE Le cifre esadecimali vanno da `0` a `f`; quindi ogni cifra può contenere uno dei 16 differenti valori.

Per semplificare le cose, gli zeri iniziali di ogni gruppo possono essere rimossi; tuttavia, ogni gruppo deve contenere almeno una cifra:

```
2001:db8:0:abcd:0:0:0:7334
```

Quando più gruppi contenenti solo zeri si susseguono direttamente uno dopo l'altro, possono essere completamente sostituiti da "::":

```
2001:db8:0:abcd::7334
```

Tuttavia, questo può accadere solo una volta in ogni indirizzo.

Prefisso IPv6

I primi 64 bit di un indirizzo IPv6 sono noti come *prefisso di routing* (*routing prefix*). Il prefisso viene utilizzato dai router per determinare a quale rete appartiene un dispositivo e, quindi, su quale percorso devono essere inviati i dati. Il subnetting avviene sempre all'interno del prefisso. Gli ISP di solito distribuiscono i prefissi /48 o /58 ai propri clienti, lasciando 16 o 8 bit per il loro subnetting interno.

Esistono tre principali tipi di prefisso in IPv6:

Global Unique Address

Il prefisso viene assegnato dai blocchi riservati agli indirizzi globali. Questi indirizzi sono validi in tutta Internet.

Unique Local Address

Non può essere instradato su Internet. Possono, tuttavia, essere instradati internamente all'interno di un'organizzazione. Questi indirizzi vengono utilizzati all'interno di una rete per garantire che i dispositivi abbiano un indirizzo anche in assenza di connessione Internet. Sono l'equivalente degli intervalli di indirizzi privati in IPv4. I primi 8 bit sono sempre `fc` o `fd`, seguiti da 40 bit generati casualmente.

Link Local Address

Sono validi solo su un particolare link. Ogni interfaccia di rete compatibile con IPv6 ha uno di questi indirizzi, che inizia con `fe80`. Questi indirizzi vengono utilizzati internamente da IPv6 per richiedere indirizzi aggiuntivi utilizzando la configurazione automatica e per trovare altri computer sulla rete utilizzando il *Neighbor Discovery Protocol*.

Identificativo dell'Interfaccia IPv6

Mentre il prefisso determina in quale rete risiede un dispositivo, l'identificativo dell'interfaccia viene utilizzato per elencare i dispositivi all'interno di una rete. Gli ultimi 64 bit in un indirizzo IPv6 formano l'identificativo dell'interfaccia, proprio come gli ultimi bit di un indirizzo IPv4.

Quando un indirizzo IPv6 viene assegnato manualmente, l'identificativo dell'interfaccia viene impostato come parte dell'indirizzo. Quando si utilizza la configurazione automatica dell'indirizzo, l'identificativo dell'interfaccia viene scelto casualmente o derivato dall'indirizzo link layer del dispositivo. Questo fa apparire una variante dell'indirizzo di link layer all'interno dell'indirizzo IPv6.

Configurazione di Indirizzi IPv6

Come per IPv4, un indirizzo IPv6 può essere assegnato sia manualmente sia automaticamente. Tuttavia, IPv6 ha due diversi tipi di configurazione automatica, DHCPv6 e *Stateless Address Autoconfiguration* (SLAAC).

SLAAC è il più semplice dei due metodi automatizzati ed è integrato nello standard IPv6. I messaggi utilizzano il nuovo *Neighbor Discovery Protocol* che consente ai dispositivi di trovarsi e richiedere informazioni su una rete. Queste informazioni vengono inviate dai router e possono contenere prefissi IPv6 che i dispositivi possono utilizzare combinandoli con un identificativo di interfaccia di loro scelta, purché l'indirizzo risultante non sia ancora in uso. I dispositivi non forniscono feedback al router sugli indirizzi effettivi che hanno creato.

DHCPv6, invece, è il DHCP aggiornato per funzionare con le modifiche di IPv6. Consente un controllo più preciso delle informazioni fornite ai *client*, per esempio consentendo ogni volta la distribuzione dello stesso indirizzo allo stesso client e l'invio di più opzioni al client rispetto a SLAAC. Con DHCPv6, i client devono ottenere il consenso esplicito di un server DHCP per utilizzare un indirizzo.

Il metodo per assegnare manualmente un indirizzo IPv6 a un'interfaccia è lo stesso di IPv4:

```
$ sudo ip addr add 2001:db8:0:abcd:0:0:0:7334/64 dev ens33
```

Per verificare che l'operazione di assegnazione sia andata a buon fine viene utilizzato lo stesso comando `ip addr show` visto in precedenza:

```
$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
25: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen
1000
```

```
    link/ether 00:0c:29:33:3b:25 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.5/24 192.168.0.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::010c:29ff:fe33:3b25/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
    inet6 2001:db8:0:abcd::7334/64 scope global
       valid_lft forever preferred_lft forever
```

Qui possiamo vedere anche l'indirizzo link-local `fe80::010c:29ff:fe33:3b25/64`.

Come per IPv4, il comando `ping` può essere utilizzato per confermare la raggiungibilità dei dispositivi anche in IPv6:

```
$ ping 2001:db8:0:abcd::1
PING 2001:db8:0:abcd::1(2001:db8:0:abcd::1) 56 data bytes
64 bytes from 2001:db8:0:abcd::1: icmp_seq=1 ttl=64 time=0.030 ms
64 bytes from 2001:db8:0:abcd::1: icmp_seq=2 ttl=64 time=0.040 ms
64 bytes from 2001:db8:0:abcd::1: icmp_seq=3 ttl=64 time=0.072 ms

--- 2001:db8:0:abcd::1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 43ms
rtt min/avg/max/mdev = 0.030/0.047/0.072/0.018 ms
```

NOTE Su alcuni sistemi Linux, `ping` non supporta IPv6. Questi sistemi hanno invece un comando `ping6` dedicato.

Per verificare nuovamente l'indirizzo link-local, puoi utilizzare ancora `ping`. Ma poiché tutte le interfacce usano il prefisso `fe80::/64`, l'interfaccia corretta deve essere specificata insieme all'indirizzo:

```
$ ping6 -c 1 fe80::010c:29ff:fe33:3b25%ens33
PING fe80::010c:29ff:fe33:3b25(fe80::010c:29ff:fe33:3b25) 56 data bytes
64 bytes from fe80::010c:29ff:fe33:3b25%ens33: icmp_seq=1 ttl=64 time=0.049 ms

--- fe80::010c:29ff:fe33:3b25 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.049/0.049/0.049/0.000 ms
```

DNS

Gli indirizzi IP sono difficili da ricordare e non sono sicuramente molto attraenti se stai cercando

di commercializzare un servizio o un prodotto. È qui che entra in gioco il *Domain Name System*. Nella sua forma più semplice, il DNS è una rubrica telefonica distribuita che mappa nomi di dominio facili da ricordare, come `example.com`, su indirizzi IP. Quando, per esempio, un utente naviga su un sito Web, immette il nome host del DNS come parte dell'URL. Il browser web invia quindi il nome del DNS a qualsiasi resolver DNS sia stato configurato. Questo *resolver* DNS scopre a sua volta l'indirizzo correlato al dominio; quindi risponde con quell'indirizzo e il browser web cerca di raggiungere il server web a quell'indirizzo IP.

I resolver che Linux utilizza per cercare i dati di DNS si trovano nel file di configurazione `/etc/resolv.conf`:

```
$ cat /etc/resolv.conf
search lpi
nameserver 192.168.0.1
```

Quando il resolver esegue la ricerca di un nome, controlla prima il file `/etc/hosts` per vedere se contiene un indirizzo per il nome richiesto. Se lo trova, restituisce quell'indirizzo e non contatta il DNS. Ciò consente agli amministratori di rete di impostare la risoluzione dei nomi senza dover configurare un server DNS completo. Ogni riga in qesto file contiene un indirizzo IP seguito da uno o più nomi:

```
127.0.0.1          localhost.localdomain   localhost
::1                localhost.localdomain   localhost
192.168.0.10       server
2001:db8:0:abcd::f server
```

Per eseguire una ricerca nel DNS, usa il comando `host`:

```
$ host learning.lpi.org
learning.lpi.org has address 208.94.166.198
```

Puoi ottenere informazioni più dettagliate usando il comando `dig`:

```
$ dig learning.lpi.org
; <<>> DiG 9.14.3 <<>> learning.lpi.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 21525
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
```

```
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 2ac55879b1adef30a93013705d3306d2128571347df8eadf (bad)
;; QUESTION SECTION:
;learning.lpi.org.      IN  A

;; ANSWER SECTION:
learning.lpi.org.   550 IN  A   208.94.166.198

;; Query time: 3 msec
;; SERVER: 192.168.0.1#53(192.168.0.1)
;; WHEN: Sat Jul 20 14:20:21 EST 2019
;; MSG SIZE  rcvd: 89
```

Qui possiamo anche vedere il nome dei tipi di record DNS, in questo caso `A` per IPv4.

Socket

Un *socket* è un punto finale di comunicazione per due programmi che parlano tra loro. Se il socket è connesso a una rete, i programmi possono essere eseguiti su diversi dispositivi, come un browser web in esecuzione sul laptop di un utente e un server web in esecuzione nel data center di un'azienda.

Esistono tre tipi principali di socket:

Socket Unix

Connettono processi in esecuzione sullo stesso dispositivo.

Socket UDP (User Datagram Protocol)

Connettono applicazioni utilizzando un protocollo veloce ma non robusto.

Socket TCP (Transmission Control Protocol)

Sono più affidabili dei socket UDP e, per esempio, confermano la ricezione dei dati.

I socket Unix possono solo connettere applicazioni in esecuzione sullo stesso dispositivo. Tuttavia, i socket TCP e UDP possono stabilire connessioni su una rete. TCP consente a un flusso di dati di arrivare sempre nell'ordine esatto in cui è stato inviato. UDP è più "invia e dimentica"; il pacchetto viene inviato, ma la sua consegna non è garantita. UDP, tuttavia, non ha l'*overhead* di TCP, rendendolo perfetto per applicazioni a bassa latenza come i videogiochi online.

Sia TCP sia UDP utilizzano delle porte per indirizzare più socket sullo stesso indirizzo IP. Mentre la porta di origine per una connessione è solitamente casuale, le porte di destinazione sono

standardizzate per uno specifico servizio. HTTP, per esempio, è solitamente ospitato sulla porta 80, mentre HTTPS viene eseguito sulla porta 443. SSH, un protocollo per accedere in modo sicuro a un sistema Linux remoto, è in ascolto sulla porta 22.

Il comando `ss` consente a un amministratore di esaminare tutti i socket su un computer Linux. Mostra tutto: dall'indirizzo di origine, all'indirizzo di destinazione e al tipo. Una delle sue migliori caratteristiche è l'uso di filtri in modo che un utente possa monitorare i socket in qualsiasi stato di connessione desideri. `ss` può essere eseguito con un insieme di opzioni oltre che un'espressione di filtro per limitare le informazioni mostrate.

Quando viene eseguito senza alcuna opzione, il comando mostra un elenco di tutti i socket stabiliti. L'opzione `-p` include informazioni sul processo che utilizza ogni socket. L'opzione `-s` mostra un riepilogo dei socket. Ci sono molte altre opzioni disponibili per questo strumento, ma le ultime principali da ricordare sono `-4` e `-6` per restringere il campo di ricerca del protocollo IP rispettivamente a IPv4 o IPv6, `-t` e `-u` per consentire all'amministratore di selezionare i socket TCP o UDP, e `-l` per mostrare solo i socket che sono in attesa di nuove connessioni.

Il seguente comando, per esempio, elenca tutti i socket TCP attualmente in uso:

```
$ ss -t
State       Recv-Q  Send-Q    Local Address:Port      Peer Address:Port
ESTAB       0       0           192.168.0.5:49412      192.168.0.1:https
ESTAB       0       0           192.168.0.5:37616      192.168.0.1:https
ESTAB       0       0           192.168.0.5:40114      192.168.0.1:https
ESTAB       0       0           192.168.0.5:54948      192.168.0.1:imap
...
```

Esercizi Guidati

1. A un ingegnere di rete viene chiesto di assegnare due indirizzi IP all'interfaccia `ens33` di un host: un indirizzo IPv4 (`192.168.10.10/24`) e un indirizzo IPv6 (`2001:0:0:abcd:0:8a2e:0370:7334/64`). Quali comandi deve usare?

2. Quali indirizzi del seguente elenco sono privati?

`192.168.10.1`	
`120.56.78.35`	
`172.16.57.47`	
`10.100.49.162`	
`200.120.42.6`	

3. Quale voce dovresti aggiungere nel file *hosts* per assegnare `192.168.0.15` a `example.com`?

4. Qual è l'effetto del seguente comando?

```
sudo ip -6 route add default via 2001:db8:0:abcd::1
```

Esercizi Esplorativi

1. Indica il tipo di record DNS utilizzato per le seguenti richieste:
   - Dati testuali
   - Ricerca inversa per indirizzo IP (Reverse IP address lookup)
   - Un dominio che non ha un proprio indirizzo e fa affidamento su un altro dominio per queste informazioni
   - Server di posta
2. Linux prevede una funzionalità chiamata *bridging*: cosa fa e come può esserti utile?
3. Quale opzione deve essere passata al comando `ss` per visualizzare tutti i socket UDP stabiliti?
4. Quale comando mostra un riepilogo di tutti i socket in esecuzione su un sistema Linux?
5. Il seguente output è stato generato dal comando dell'esercizio precedente. Quanti socket TCP e UDP sono attivi?

```
Total: 978 (kernel 0)
TCP:   4 (estab 0, closed 0, orphaned 0, synrecv 0, timewait 0/0), ports 0

Transport Total     IP        IPv6
*     0         -         -
RAW   1         0         1
UDP   7         5         2
TCP   4         3         1
INET      12        8         4
FRAG      0         0         0
```

Sommario

Questa lezione tratta il collegamento in rete di un computer Linux. Per prima cosa abbiamo imparato a conoscere i vari livelli di rete:

- Il link layer che collega direttamente i dispositivi;
- Il netwok layer che prevede il routing tra le reti e uno spazio di indirizzi globale;
- L'application Layer in cui le applicazioni si connettono tra loro.

Abbiamo visto come IPv4 e IPv6 vengono utilizzati per indirizzare i singoli computer e come TCP e UDP enumerano i socket utilizzati dalle applicazioni per connettersi tra loro. Abbiamo anche appreso come viene utilizzato il DNS per risolvere i nomi in indirizzi IP.

Comandi utilizzati negli esercizi:

`dig`
: Interroga il DNS e fornisce informazioni dettagliate sulle domande e sulle risposte.

`host`
: Interroga il DNS e fornisce un output sintetico.

`ip`
: Configura la rete su Linux, comprese le interfacce di rete, gli indirizzi e il routing.

`ping`
: Testa la connettività di un dispositivo remoto.

`ss`
: Mostra informazioni sui socket.

Risposte agli Esercizi Guidati

1. A un ingegnere di rete viene chiesto di assegnare due indirizzi IP all'interfaccia `ens33` di un host: un indirizzo IPv4 (`192.168.10.10/24`) e un indirizzo IPv6 (`2001:0:0:abcd:0:8a2e:0370:7334/64`). Quali comandi deve usare?

```
sudo ip addr add 192.168.10.10/24 dev ens33
sudo ip addr add 2001:0:0:abcd:0:8a2e:0370:7334/64 dev ens33
```

2. Quali indirizzi del seguente elenco sono privati?

`192.168.10.1`	X
`120.56.78.35`	
`172.16.57.47`	X
`10.100.49.162`	X
`200.120.42.6`	

3. Quale voce dovresti aggiungere nel file *hosts* per assegnare `192.168.0.15` a `example.com`?

```
192.168.0.15  example.com
```

4. Qual è l'effetto del seguente comando?

```
sudo ip -6 route add default via 2001:db8:0:abcd::1
```

   Aggiunge una default route nella tabella di routing che invia tutto il traffico IPv6 al router con indirizzo interno `2001:db8:0:abcd::1`.

Risposte agli Esercizi Esplorativi

1. Indica il tipo di record DNS utilizzato per le seguenti richieste:
   - Dati testuali

     TXT

   - Ricerca inversa per indirizzo IP (Reverse IP address lookup)

     PTR

   - Un dominio che non ha un proprio indirizzo e fa affidamento su un altro dominio per queste informazioni

     CNAME

   - Server di posta

     MX

2. Linux prevede una funzionalità chiamata *bridging*: cosa fa e come può esserti utile?

   Un bridge collega più interfacce di rete. Tutte le interfacce connesse a un bridge possono comunicare come se fossero connesse alla stessa rete link layer: tutti i dispositivi utilizzano indirizzi IP della stessa sottorete e non richiedono un router per connettersi tra loro.

3. Quale opzione deve essere passata al comando `ss` per visualizzare tutti i socket UDP stabiliti?

   L'opzione `-u` mostra tutti i socket UDP stabiliti.

4. Quale comando mostra un riepilogo di tutti i socket in esecuzione su un sistema Linux?

   Il comando `ss -s` mostra un riepilogo di tutti i socket.

5. Il seguente output è stato generato dal comando dell'esercizio precedente. Quanti socket TCP e UDP sono attivi?

```
Total: 978 (kernel 0)
TCP:   4 (estab 0, closed 0, orphaned 0, synrecv 0, timewait 0/0), ports 0

Transport Total     IP        IPv6
*     0         -         -
RAW   1         0         1
```

```
UDP   7         5         2
TCP   4         3         1
INET      12        8         4
FRAG      0         0         0
```

I socket TCP e UDP attivi sono 11.

Linux Professional Institute

Argomento 5: Sicurezza e Permessi sui File

5.1 Sicurezza di Base e Identificazione dei Tipi di Utente

Obiettivi LPI di riferimento

Linux Essentials version 1.6, Exam 010, Objective 5.1

Peso

2

Aree di Conoscenza Chiave

- Root e utenti standard
- Utenti di sistema

Di seguito è riportato un elenco parziale dei file, dei termini e dei comandi utilizzati

- `/etc/passwd`, `/etc/shadow`, `/etc/group`
- `id`, `last`, `who`, `w`
- `sudo`, `su`

5.1 Lezione 1

Certificazione:	Linux Essentials
Versione:	1.6
Argomento:	5 Sicurezza e Permessi sui File
Obiettivo:	5.1 Sicurezza di Base e Identificazione dei Tipi di Utente
Lezione:	1 di 1

Introduzione

Questa lezione tratta la terminologia di base degli account, il controllo degli accessi e la sicurezza dei sistemi Linux locali, gli strumenti della *Command Line Interface* (CLI) in un sistema Linux per i controlli degli accessi per la sicurezza di base e i file principali per gli account utente e di gruppo, inclusi quelli utilizzati per l'escalation dei privilegi di base.

La sicurezza di base nei sistemi Linux deriva dai controlli d'accesso Unix che, nonostante abbiano quasi cinquant'anni, sono molto efficaci in confronto ad alcuni popolari sistemi operativi di livello *consumer* molto più recenti. Alcuni altri popolari sistemi operativi basati su Unix tendono a "prendersi delle libertà" in termini di "facilità di accesso", ma Linux no.

I moderni ambienti desktop Linux e le relative interfacce semplificano la creazione e la gestione degli utenti e spesso automatizzano l'assegnazione dei controlli di accesso quando un utente effettua il login, per esempio al display, all'audio e ad altri servizi, senza richiedere praticamente alcun intervento manuale da parte dell'amministratore di sistema. Tuttavia è importante comprendere i concetti di base del sistema operativo Linux sottostante.

Account

La sicurezza comprende molti concetti: uno dei più comuni è il concetto generale di controllo degli accessi. Prima di poter affrontare i controlli di accesso ai file come *ownership* e autorizzazioni, è necessario comprendere i concetti base relativi agli *account* utente Linux, che sono suddivisi in diversi tipi.

In un sistema Linux ogni utente è associato a un account che oltre alle informazioni di accesso (come nome utente e password) definisce anche come e dove l'utente può interagire con il sistema. Privilegi e controlli di accesso definiscono i "confini" entro i quali ogni utente può operare.

Identificativi (UID/GID)

Gli *identificativi utente* e *di gruppo* (UID - User Identifier/GID - Group Identifier) sono i riferimenti numerici di base agli account. Le prime implementazioni prevedevano numeri interi limitati a 16 bit (valori da 0 a 65535), ma i sistemi moderni (del 21° secolo) supportano UID e GID a 64 bit. Utenti e gruppi vengono enumerati in modo indipendente: quindi lo stesso ID può rappresentare sia un utente sia un gruppo.

Ogni utente ha non solo un UID, ma anche un *GID primario*. Il GID primario per un utente può essere univoco solo per quell'utente e potrebbe non essere utilizzato da altri utenti. Tuttavia, questo gruppo potrebbe anche essere un gruppo condiviso da numerosi utenti. Oltre a questi gruppi primari, ogni utente può essere membro anche di altri gruppi.

Sui sistemi Linux, per impostazione predefinita, ogni utente viene assegnato a un gruppo con lo stesso nome del proprio nome utente, e lo stesso GID del proprio UID. Per esempio, se si crea un nuovo utente chiamato `newuser`, per impostazione predefinita anche il suo gruppo di default sarà `newuser`.

L'Account Superuser

Su Linux l'account *superuser* è `root`, che ha sempre UID 0. Talvolta il superuser è chiamato *amministratore di sistema* e ha accesso e controllo illimitati sul sistema, inclusi gli altri utenti.

Il gruppo predefinito per il superuser ha GID `0` e anch'esso è chiamato `root`. La directory home del superuser è una directory di primo livello dedicata, `/root`, accessibile solo dall'utente `root` stesso.

Account Utenti Standard

Tutti gli account diversi da `root` sono tecnicamente account utenti ordinari, ma su un sistema Linux il termine colloquiale *account utente* spesso si riferisce a un account utente "ordinario" (non privilegiato). In genere hanno le seguenti proprietà, con alcune eccezioni:

- UID che iniziano da 1000 (4 cifre), sebbene in alcuni sistemi *legacy* possano iniziare da 500.
- Una determinata directory home, solitamente una sottodirectory di `/home`, a seconda della configurazione del sistema.
- Una determinata shell di login. In Linux la shell predefinita è solitamente la *Bourne Again Shell* (`/bin/bash`), sebbene possano essere disponibili anche altre.

Se un account utente non ha una shell valida nei propri attributi, l'utente non sarà in grado di aprire una shell interattiva. Di solito `/sbin/nologin` viene usato per indicare una shell non valida. Questo può essere utile nel caso in cui l'utente venga autenticato solo per servizi diversi dalla console o dall'accesso SSH, per esempio solo per l'accesso FTP sicuro (`sftp`).

NOTE Per evitare confusione il termine *account utente* in futuro verrà usato solo per indicare gli account utenti standard od ordinari. Per esempio, il termine *account di sistema* verrà utilizzato per fare riferimento a un account utente Linux appartenenete al tipo di account utente di sistema.

Account di Sistema

Gli *account di sistema* vengono generalmente creati durante l'installazione del sistema. Questi sono destinati a strutture (*facility*), programmi e servizi che non verranno eseguiti come superuser. In un mondo ideale, sarebbero tutte facility del sistema operativo.

Gli account di sistema variano, ma i loro attributi includono:

- Gli UID sono generalmente inferiori a 100 (2 cifre) o vanno da 500 a 1000 (3 cifre).
- O *nessuna* directory home dedicata o una directory che di solito non si trova sotto `/home`.
- Nessuna shell di login valida (tipicamente `/sbin/nologin`), con rare eccezioni.

La maggior parte degli account di sistema su Linux non effettuerà mai il login e quindi non necessita di una shell definita nei propri attributi. Molti processi che hanno come come owner e vengono eseguiti dagli account di sistema vengono biforcati nel proprio ambiente dal gestore di sistema, in esecuzione con l'account di sistema specificato. Questi account di solito hanno privilegi limitati o, il più delle volte, *nessun* privilegio.

NOTE Per la certificazione LPI Linux Essentials, gli account di sistema sono UID < 1000,

con UID (e GID) a 2 o 3 cifre.

In generale gli account di sistema *non* dovrebbero avere una shell di login valida. Il contrario costituirebbe un problema di sicurezza nella maggior parte dei casi.

Account di Servizio

Gli *account di servizio* vengono generalmente creati quando vengono installati e configurati i servizi. Similmente agli account di sistema, sono destinati a strutture (facility), programmi e servizi che non vengono eseguiti come superuser.

In molta documentazione gli account di sistema e di servizio sono simili e spesso interscambiabili. Ciò include la posizione delle directory home, che, se definite, tipicamente si trovano al di fuori di `/home` (gli account di servizio hanno spesso maggiori probabilità di averne una rispetto agli account di sistema) e nessuna shell di login valida. Sebbene non esista una definizione rigorosa, la differenza principale tra gli account di sistema e di servizio sta negli UID/GID.

Account di sistema

UID/GID < 100 (2 cifre) o < 500-1000 (3 cifre)

Account di servizio

UID/GID > 1000 (4+ cifre), ma non un account utente “standard” o “ordinario”, un account per servizi, con un UID/GID > 1000 (4+ cifre)

Alcune distribuzioni Linux hanno ancora account di servizio pre-riservati con UID < 100 e potrebbero anche essere considerati account di sistema, anche se non vengono creati durante l’installazione. Per esempio, nelle distribuzioni Linux basate su Fedora (incluso Red Hat), l’utente per il server Web Apache ha UID (e GID) 48, chiaramente un account di sistema, nonostante abbia una home directory (di solito in `/usr/share/httpd` o `/var/www/html/`).

NOTE Per la certificazione LPI Linux Essentials, gli account di sistema sono UID < 1000 e gli account utente ordinari sono UID > 1000. Poiché gli account utente ordinari sono > 1000, questi UID possono includere anche account di servizio.

Shell di Login e Directory Home

Alcuni account hanno una shell di login, mentre altri no per motivi di sicurezza, in quanto non richiedono un accesso interattivo. La shell di login predefinita nella maggior parte delle distribuzioni Linux è la *Bourne Again Shell*, o `bash`, ma potrebbero essere disponibili anche altre shell, come la shell C (`csh`), la shell Korn (`ksh`) o la shell Z (`zsh`), giusto per nominarne alcune.

Un utente può cambiare la sua shell di login utilizzando il comando `chsh`. Per impostazione

predefinita, il comando viene eseguito in modalità interattiva e visualizza un prompt che chiede quale shell debba essere utilizzata. La risposta dovrebbe essere il percorso completo del binario della shell, come mostrato qui sotto:

```
$ chsh

Changing the login shell for emma
Enter the new value, or press ENTER for the default
    Login Shell [/bin/bash]: /usr/bin/zsh
```

Puoi anche eseguire il comando in modalità non interattiva, con l'opzione `-s` seguita dal percorso del binario, in questo modo:

```
$ chsh -s /usr/bin/zsh
```

La maggior parte degli account ha una directory home definita. In Linux, questa è solitamente l'unico posto in cui un account utente ha garantito l'accesso in scrittura, con alcune eccezioni (per esempio aree temporanee del file system). Tuttavia, per motivi di sicurezza, alcuni account sono impostati appositamente per non avere alcun accesso in scrittura anche nella propria directory home.

Ottenere Informazioni Sugli Utenti

Elencare le informazioni di base degli utenti è una pratica quotidiana comune su un sistema Linux. In alcuni casi gli utenti dovranno cambiare utente e ottenere privilegi più alti per portare a termine attività privilegiate.

Anche gli utenti ordinari hanno la possibilità di elencare gli attributi e gli accessi dalla Command Line, utilizzando i comandi di seguito riportati. Le informazioni di base, in un contesto limitato, non costituiscono un'operazione privilegiata.

Per elencare sulla Command Line le informazioni correnti di un utente basta un semplice comando di due lettere: `id`. L'output varierà in base al tuo ID di login:

```
$ id
uid=1024(emma) gid=1024(emma) 1024(emma),20(games),groups=10240(netusers),20480(netadmin)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Nell'elenco precedente, l'utente (`emma`) possiede una serie di identificativi che possono essere

suddivisi come segue:

- `1024` = User ID (UID), seguito dal nome utente (nome comune o nome di login) tra parentesi.
- `1024` = l'ID del gruppo *primario* (GID), seguito dal nome del gruppo (nome comune) tra parentesi.
- Un elenco di GID aggiuntivi (nomi di gruppo) a cui appartiene anche l'utente.

Per elencare l'ultima volta che gli utenti si sono loggati sul sistema puoi usare il comando `last`:

```
$ last
emma     pts/3        ::1              Fri Jun 14 04:28   still logged in
reboot   system boot  5.0.17-300.fc30. Fri Jun 14 04:03   still running
reboot   system boot  5.0.17-300.fc30. Wed Jun  5 14:32 - 15:19  (00:46)
reboot   system boot  5.0.17-300.fc30. Sat May 25 18:27 - 19:11  (00:43)
reboot   system boot  5.0.16-100.fc28. Sat May 25 16:44 - 17:06  (00:21)
reboot   system boot  5.0.9-100.fc28.x Sun May 12 14:32 - 14:46  (00:14)
root     tty2                         Fri May 10 21:55 - 21:55  (00:00)
    ...
```

Le informazioni elencate nelle colonne possono variare, ma alcune voci degne di nota dell'elenco precedente sono:

- Un utente (`emma`) ha effettuato il login tramite la rete (pseudo TTY `pts/3`) ed è ancora loggato.
- Viene elencata l'ora del processo di avvio corrente, insieme al kernel. Nell'esempio sopra riportato, circa 25 minuti prima che l'utente abbia effettuato l'accesso.
- Il superuser (`root`) ha effettuato il login tramite una console virtuale (TTY `tty2`), sommariamente, a metà maggio.

Una variante del comando `last` è il comando `lastb`, che elenca tutti gli ultimi tentativi di login falliti.

I comandi `who` e `w` elencano solamente i login attivi sul sistema:

```
$ who
emma  pts/3        2019-06-14 04:28 (::1)

$ w
 05:43:41 up  1:40,  1 user,  load average: 0.25, 0.53, 0.51
USER     TTY        LOGIN@   IDLE   JCPU   PCPU WHAT
emma  pts/3     04:28    1:14m  0.04s  0.04s -bash
```

Alcune informazioni sono mostrate da entrambi i comandi. Per esempio, un utente (`emma`) ha effettuato il login con uno pseudo-dispositivo TTY (`pts/3`) e l'ora del login è 04:28.

Il comando `w` mostra ulteriori informazioni, incluse le seguenti:

- L'ora corrente e da quanto tempo il sistema è attivo
- Il numero di utenti collegati
- Il *carico medio* degli ultimi 1, 5 e 15 minuti

Vengono inoltre visualizzate informazioni aggiuntive per ogni sessione utente attiva:

- Tempi totali di utilizzo della CPU (Select, total CPU utilization times) (`IDLE`, `JCPU` e `PCPU`)
- Il processo corrente (`-bash`). Il tempo di utilizzo totale della CPU da parte di quel processo è l'ultimo elemento (`PCPU`).

Entrambi i comandi hanno ulteriori opzioni per elencare varie informazioni aggiuntive.

Cambiare Utente e Aumentare i Privilegi

In un mondo ideale, gli utenti non avrebbero mai bisogno di aumentare i privilegi per completare le proprie attività. Il sistema "funzionerebbe" sempre e tutto sarebbe configurato per i diversi tipi di accesso.

Fortunatamente per noi, Linux funziona in questo modo per la maggior parte degli utenti che non sono amministratori di sistema, nonostante segua sempre il modello di sicurezza con *privilegi minimi*.

Tuttavia, esistono comandi che consentono l'escalation dei privilegi quando necessario. Due dei più importanti sono `su` e `sudo`.

Attualmente, nella maggior parte dei sistemi Linux, il comando `su` viene utilizzato solo per aumentare i privilegi a *root*, che è l'utente predefinito se non viene specificato un nome utente dopo il comando. Sebbene possa essere utilizzato per passare a un altro utente, non è certamente una buona pratica: gli utenti devono effettuare il login da un altro sistema, sulla rete, o dalla console fisica o dal terminale del sistema.

```
emma ~$ su -
Password:
root ~#
```

Dopo aver inserito la password del superuser (`root`), l'utente ottiene la sua shell (nota il `#` alla fine del prompt dei comandi) e diventa, a tutti gli effetti, il superuser (`root`).

La condivisione delle password è una pessima pratica di sicurezza; quindi, in un moderno sistema Linux, dovrebbe esserci pochissimo bisogno, se presente, del comando `su`.

Il simbolo del dollaro (`$`) dovrebbe terminare il prompt della Command Line di una shell utente non privilegiata, mentre il simbolo cancelletto (#) dovrebbe terminare il prompt della Command Line della shell del superuser (`root`). È altamente raccomandato *non* cambiare *mai* il carattere finale di qualsiasi prompt da questo standard "universalmente accettato", poiché questa nomenclatura è usata nei materiali di apprendimento, inclusi questi.

WARNING

Non passare mai al superuser (`root`) senza passare il parametro della shell di login (`-`). A meno che non sia esplicitamente indicato diversamente dal SO o dal fornitore del software, quando `su` è necessario, dovresti eseguire sempre `su -` con pochissime eccezioni. Gli ambienti utente possono causare modifiche indesiderate alla configurazione e problemi se utilizzati in modalità con privilegi completi come superuser.

Qual è il problema più grande nell'usare `su` per passare al superuser (`root`)? Se la sessione di un utente ordinario è stata compromessa, la password del superuser (`root`) potrebbe essere individuata. È qui che entra in gioco "Switch User Do" (o "Superuser Do"):

```
$ cat /sys/devices/virtual/dmi/id/board_serial
cat: /sys/devices/virtual/dmi/id/board_serial: Permission denied

$ sudo cat /sys/devices/virtual/dmi/id/board_serial
[sudo] password for emma:
/6789ABC/
```

Nell'esempio precedente, l'utente sta tentando di individuare il numero di serie della propria scheda di sistema. Tuttavia, l'autorizzazione viene negata, poiché tali informazioni sono contrassegnate come privilegiate.

Tuttavia, usando `sudo`, l'utente inserisce la propria password per autenticarsi. Se è stato autorizzato nella configurazione di `sudoers` a eseguire quel comando con privilegi, con le opzioni consentite, funzionerà.

TIP

Per impostazione predefinita, il primo comando `sudo` autorizzato autenticherà i successivi comandi `sudo` per un certo periodo di tempo (molto breve). Questo è configurabile dall'amministratore di sistema.

File di Controllo degli Accessi

Quasi tutti i sistemi operativi hanno una serie di posizioni specifiche in cui memorizzare i controlli di accesso. In Linux questi sono tipicamente file di testo nella directory `/etc`, che è dove dovrebbero essere memorizzati i file di configurazione del sistema. Per impostazione predefinita, questa directory è leggibile da ogni utente sul sistema, ma scrivibile solo da `root`.

I file principali relativi agli account utente, agli attributi e al controllo degli accessi sono:

`/etc/passwd`

Questo file memorizza le informazioni di base sugli utenti di sistema, inclusi UID e GID, la directory home, la shell, etc. Nonostante il nome, qui non vengono memorizzate password.

`/etc/group`

Questo file memorizza le informazioni di base su tutti i gruppi di utenti nel sistema, come il nome del gruppo, il GID e i membri.

`/etc/shadow`

Qui è dove vengono memorizzate le password degli utenti. Sono crittografate (sottoposte a *hashing*) per ragioni di sicurezza.

`/etc/gshadow`

Questo file memorizza informazioni più dettagliate sui gruppi, inclusa una password crittografata (sottoposta a hashing) che consente agli utenti di diventare temporaneamente membri del gruppo, un elenco di utenti che possono diventare membri del gruppo in qualsiasi momento e un elenco di amministratori del gruppo.

WARNING Questi file non sono stati progettati per essere modificati direttamente e questo non dovrebbe mai accadere. Questa lezione tratta solamente le informazioni memorizzate in questi file e non la loro modifica.

Per impostazione predefinita ogni utente può entrare in `/etc` e leggere i file `/etc/passwd` ed `/etc/group`. Inoltre, per impostazione predefinita, nessun utente, eccetto `root`, può leggere i file `/etc/shadow` ed `/etc/gshadow`.

Esistono anche dei file che hanno a che fare con l'escalation dei privilegi di base sui sistemi Linux, come, per esempio, accade con i comandi `su` e `sudo`. Per impostazione predefinita, questi sono accessibili solo dall'utente `root`.

`/etc/sudoers`

Questo file controlla chi e come può usare il comando `sudo`.

`/etc/sudoers.d`

Questa directory può contenere file che integrano le configurazioni del file `sudoers`.

Per l'esame della certificazione LPI Linux Essentials, è sufficiente conoscere il percorso e il nome del file di configurazione predefinito di sudo: `/etc/sudoers`. La sua configurazione va oltre lo scopo di questi materiali.

WARNING

Anche se `/etc/sudoers` è un file di testo, non dovrebbe mai essere modificato direttamente. Se sono necessarie modifiche al suo contenuto, dovrebbero essere fatte utilizzando l'utility `visudo`.

/etc/passwd

Il file `/etc/passwd` è comunemente chiamato "file delle password". Ogni riga contiene più campi sempre delimitati dal carattere due punti (`:`). Nonostante il nome, l'hash unidirezionale (one-way hash) delle password oggi non è più memorizzato in questo file.

La tipica sintassi di una riga in questo file è la seguente:

```
USERNAME:PASSWORD:UID:GID:GECOS:HOMEDIR:SHELL
```

Dove:

`USERNAME`

Il nome utente o di login (login name), come `root`, `nobody`, `emma`.

`PASSWORD`

Vecchia posizione dell'hash della password. È quasi sempre `x`, a indicare che la password è memorizzata nel file `/etc/shadow`.

`UID`

User ID (UID), come `0`, `99`, `1024`.

`GID`

Group ID predefinito (GID), come `0`, `99`, `1024`.

`GECOS`

Un elenco CSV di informazioni sull'utente, inclusi nome, posizione, numero di telefono. Per esempio: `Emma Smith,42 Douglas St,555.555.5555`

`HOMEDIR`

Percorso della directory home dell'utente, come `/root`, `/home/emma`, etc.

`SHELL`

La shell predefinita per questo utente, come `/bin/bash`, `/sbin/nologin`, `/bin/ksh`, etc.

Per esempio, la riga seguente descrive l'utente `emma`:

```
emma:x:1000:1000:Emma Smith,42 Douglas St,555.555.5555:/home/emma:/bin/bash
```

Comprendere il Campo GECOS

Il campo GECOS contiene tre (3) o più campi, delimitati da una virgola (`,`), ovvero un elenco di *valori separati da virgola* (CSV - *Comma Separated Values*). Sebbene non esista uno standard imposto, i campi sono generalmente nel seguente ordine:

```
NAME,LOCATION,CONTACT
```

Dove:

`NAME`

È il "nome completo" (*Full Name*) dell'utente o il "nome del software" (*Software Name*) nel caso di un account di servizio.

`LOCATION`

Di solito è l'ubicazione fisica dell'utente all'interno di un edificio, il numero della stanza o l'ufficio o la persona da contattare nel caso di un account di servizio.

`CONTACT`

Elenca le informazioni di contatto come il numero di telefono di casa o dell'ufficio.

Campi aggiuntivi possono includere ulteriori informazioni di contatto, come un numero di casa (home number) o un indirizzo e-mail. Per modificare le informazioni nel campo GECOS, usa il comando `chfn` e rispondi alle domande, come nell'esempio sotto riportato. Se non viene fornito alcun nome utente dopo il comando, vengono cambiate le informazioni dell'utente corrente:

```
$ chfn

Changing the user information for emma
Enter the new value, or press ENTER for the default
```

```
    Full Name: Emma Smith
    Room Number []: 42
    Work Phone []: 555.555.5555
    Home Phone []: 555.555.6666
```

/etc/group

Il file `/etc/group` contiene campi, sempre delimitati da due punti (`:`), che memorizzano le informazioni di base sui gruppi nel sistema. A volte è chiamato “file dei gruppi”. La sintassi per ogni riga è la seguente:

```
NAME:PASSWORD:GID:MEMBERS
```

Dove:

`NAME`
: È il nome del gruppo, come `root`, `users`, `emma`, etc.

`PASSWORD`
: Vecchia posizione di un eventuale hash della password del gruppo. Quasi sempre `x`, a indicare che la password (se definita) è memorizzata nel file `/etc/gshadow`.

`GID`
: Group ID (GID), come `0`, `99`, `1024`.

`MEMBERS`
: Un elenco, separato da virgole, di nomi utente che sono membri del gruppo, come `jsmith,emma`.

L’esempio seguente mostra una riga contenente informazioni sul gruppo `students`:

```
students:x:1023:jsmith,emma
```

Non è necessario che un utente sia elencato nel campo dei membri quando il gruppo è il gruppo principale di quell’utente. Se un utente è elencato, allora è ridondante, cioè, non vi è alcun cambiamento nel funzionamento, che sia elencato o meno.

NOTE L’uso delle password per i gruppi esula dallo scopo di questa lezione. Tuttavia, se presente, l’hash della password è memorizzato nel file `/etc/gshadow`. Anche questo esula dallo scopo di questa lezione.

/etc/shadow

Il seguente elenco mostra gli attributi memorizzati nel file `/etc/shadow`, comunemente indicato come *file shadow*. Il file contiene un elenco di campi sempre delimitati da due punti (`:`). Sebbene abbia molti campi, la maggior parte di essi non rientra nell'ambito di questa lezione, a eccezione dei primi due.

La sintassi di base di una riga in questo file è la seguente:

```
USERNAME:PASSWORD:LASTCHANGE:MINAGE:MAXAGE:WARN:INACTIVE:EXPDATE
```

Dove:

`USERNAME`

Il nome utente (uguale a quello in `/etc/passwd`), come `root`, `nobody`, `emma`.

`PASSWORD`

L'hash unidirezionale (*one-way hash*) della password, incluso il salt precedente. Per esempio: `!!`, `!$1$01234567$ABC...`, `$6$012345789ABCDEF$012...`.

`LASTCHANGE`

Data dell'ultima modifica della password, espressa in giorni trascorsi dall'"Epoca", come per esempio `17909`.

`MINAGE`

Durata minima della password in giorni.

`MAXAGE`

Durata massima della password in giorni.

`WARN`

Periodo di avviso prima della scadenza della password, espresso in giorni.

`INACTIVE`

Durata massima della password dopo la scadenza, espressa in giorni.

`EXPDATE`

Data di scadenza della password, in giorni trascorsi dall'"Epoca".

Nel seguente esempio puoi vedere una riga del file `/etc/shadow`. Nota che alcuni valori, come `INACTIVE` ed `EXPDATE` non sono definiti.

```
emma:$6$nP532JDDogQYZF8I$bjFNh9eT1xpb9/n6pmjlIwgu7hGjH/eytSdttbmVv0MlyTMFgBIXESFNUmTo9EGxxH1
OT1HGQzR0so4n1npbE0:18064:0:99999:7:::
```

L'"Epoca" di un sistema POSIX è la mezzanotte (0000), Universal Coordinate Time (UTC), di giovedì 1 gennaio 1970. La maggior parte delle date e degli orari POSIX sono espresse in secondi a partire dall'"Epoca" o nel caso del file `/etc/shadow`, in giorni trascorsi dall'"Epoca".

NOTE Il file shadow è stato progettato per essere leggibile solo dal superuser e da selezionati servizi di autenticazione del sistema di base che controllano l'hash unidirezionale della password al login o in un altro momento di autenticazione.

Sebbene esistano diverse soluzioni di autenticazione, il metodo base di memorizzazione delle password è la *funzione di hash* unidirezionale. Questa viene fatta in modo tale che la password non venga mai memorizzata in chiaro su un sistema, poiché la funzione di hashing non è reversibile. Puoi trasformare una password in un hash, ma (idealmente) non è possibile trasformare un hash in una password.

Al massimo, è richiesto un metodo di forza bruta per eseguire l'hash di tutte le combinazioni di una password, fino a quando una non corrisponde. Per cercare di limitare il problema che l'hash di una password venga violato su un sistema, i sistemi Linux utilizzano un *salt* casuale per ogni hash della password di un utente. Quindi l'hash di una password utente su un sistema Linux generalmente non sarà lo stesso di un altro sistema Linux, anche se la password è la stessa.

Nel file `/etc/shadow`, la password può assumere diverse forme, tra cui le seguenti:

`!!`

Questo significa un account "disabilitato" (nessuna autenticazione possibile), senza hash della password memorizzato.

`!$1$01234567$ABC…`

Un account "disabilitato" (a causa del punto esclamativo iniziale), con una precedente funzione di hash, hash salt e hash string memorizzati.

`$1$0123456789ABC$012…`

Un account "abilitato", con una funzione di hash, hash salt e hash string memorizzati.

La funzione di hash, l'hash salt e l'hash string sono preceduti e delimitati dal simbolo dollaro (`$`). La lunghezza dell'hash salt deve essere compresa tra otto e sedici (8-16) caratteri. Ecco tre dei più comuni esempi:

`$1$01234567$ABC...`

Una funzione hash MD5 (`1`), con una lunghezza hash di otto caratteri.

`$5$01234567ABCD$012...`

Una funzione hash SHA256 (`5`), con una lunghezza hash di dodici caratteri.

`$6$01234567ABCD$012...`

Una funzione hash SHA512 (`6`), con una lunghezza hash di dodici caratteri.

NOTE

La funzione hash MD5 è considerata crittograficamente insicura per il livello di ASIC odierno (dal 2010 in poi) e persino per le prestazioni SIMD di elaborazione generale. Per esempio, i Federal Information Processing Standards (FIPS) degli Stati Uniti non consentono l'utilizzo di MD5 per alcuna funzione crittografica, solo per aspetti molto limitati di validazione, ma non per l'integrità delle firme digitali o scopi simili.

Dal punto di vista degli obiettivi e dell'esame LPI Linux Essentials, è sufficiente comprendere che l'hash della password di un utente locale è memorizzato nel file `/etc/shadow` che può essere letto solamente da selezionati servizi di autenticazione o modificato dal superuser utilizzando altri comandi.

Esercizi Guidati

1. Considera il seguente output del comando `id`:

   ```
   $ id emma
   uid=1000(emma) gid=1000(emma)
   groups=1000(emma),4(adm),5(tty),10(uucp),20(dialout),27(sudo),46(plugdev)
   ```

   In quali file sono memorizzati i seguenti attributi?

UID e GID	
Gruppi	

   - Inoltre, in quale file è memorizzata la password dell'utente?

2. Quale dei seguenti tipi di crittografia viene utilizzato, per impostazione predefinita, per memorizzare le password in locale su un sistema Linux?
   - Asimmetrica
   - Funzione di Hash unidirezionale (One-way Hash)
   - Simmetrica
   - ROT13

3. Se un account ha un ID utente (UID - User ID) inferiore a 1000, che tipo di account è?

4. Come puoi ottenere l'elenco dei login attivi sul tuo sistema e anche il loro numero?

5. Utilizzando il comando `grep`, abbiamo ottenuto le informazioni sull'utente `emma` riportate qui sotto:

   ```
   $ grep emma /etc/passwd
   emma:x:1000:1000:Emma Smith,42 Douglas St,555.555.5555,:/home/emma:/bin/ksh
   ```

   Completa gli spazi vuoti della tabella con le informazioni corrette utilizzando l'output del

comando precedente.

Nome Utente	
Password	
UID	
GID Primario	
GECOS	
Directory Home	
Shell	

Esercizi Esplorativi

1. Confronta i risultati di `last` con quelli di `w` e `who`. Quali informazioni mancano a ciascuno dei comandi rispetto agli altri?

2. Prova a eseguire i comandi `who` e `w -his`.
   - Quali informazioni sono state rimosse dall'output del comando `w` con le opzioni "no header" (`-h`) e "short" (`-s`)?
   - Quali informazioni sono state aggiunte all'output del comando `w` con l'opzione "ip address" (`-i`)?

3. Qual è il file che memorizza l'hash unidirezionale della password di un account utente?

4. Quale file contiene l'elenco dei gruppi di cui è membro un account utente? Quale logica potrebbe essere utilizzata per stilarlo?

5. Per impostazione predefinita, uno o più (1+) dei seguenti file non sono leggibili da utenti ordinari senza privilegi. Quali sono?
   - `/etc/group`
   - `/etc/passwd`
   - `/etc/shadow`
   - `/etc/sudoers`

6. Come puoi cambiare, in modalità non interattiva, la shell di login dell'utente corrente nella Shell Korn (`/usr/bin/ksh`)?

7. Perché la directory home dell'utente `root` non si trova nella directory `/home`?

Sommario

In questa lezione abbiamo trattato i database degli utenti e dei gruppi di Linux. Abbiamo appreso le proprietà più importanti di utenti e gruppi, inclusi i loro nomi e i loro ID numerici. Abbiamo anche esaminato come funziona l'hashing delle password su Linux e come gli utenti vengono assegnati ai gruppi.

Tutte queste informazioni sono memorizzate nei quattro file di seguito riportati, che forniscono i controlli di accesso più semplici per la sicurezza locale su un sistema Linux:

`/etc/passwd`
: Tutti gli attributi POSIX degli account utente a livello locale del sistema, a eccezione degli hash delle password, leggibili da tutti.

`/etc/group`
: Tutti gli attributi POSIX dei gruppi a livello locale del sistema, leggibili da tutti.

`/etc/shadow`
: Tutti gli hash delle password degli utenti a livello locale del sistema (e le informazioni sulla scadenza), non leggibili da chiunque (solo da processi selezionati).

`/etc/sudoers`
: Tutte le informazioni/permessi sull'escalation dei privilegi a livello locale del sistema tramite il comando `sudo`.

In questa lezione sono stati discussi i seguenti comandi:

`id`
: Elenca gli ID utente e di gruppo reali (o effettivi).

`last`
: Elenca gli utenti che hanno effettuato il login per ultimi.

`who`
: Elenca gli utenti che sono attualmente connessi.

`w`
: Simile a `who` ma con informazioni di contesto aggiuntive.

`su`
: Passa a un altro utente con una shell di login o esegue comandi come quell'utente, passando la

sua password.

sudo

Switch User (o Superuser) Do: se autorizzato, l'utente corrente inserisce la propria password (se richiesta) per scalare i privilegi.

chsh

Cambia la shell di un utente.

chfn

Modifica le informazioni dell'utente nel campo GECOS.

Risposte agli Esercizi Guidati

1. Considera il seguente output del comando `id`:

   ```
   $ id emma
   uid=1000(emma) gid=1000(emma)
   groups=1000(emma),4(adm),5(tty),10(uucp),20(dialout),27(sudo),46(plugdev)
   ```

   In quali file sono memorizzati i seguenti attributi?

UID e GID	`/etc/passwd`
Gruppi	`/etc/group`

   - Inoltre, in quale file è memorizzata la password dell'utente?

     La password crittografata (sottoposta a hashing) dell'utente è memorizzata in `/etc/shadow`.

2. Quale dei seguenti tipi di crittografia viene utilizzato, per impostazione predefinita, per memorizzare le password in locale su un sistema Linux?

   Per impostazione predefinita, per memorizzare le password viene utilizzata una funzione di hash unidirezionale (one-way hash).

3. Se un account ha un ID utente (UID - User ID) inferiore a 1000, che tipo di account è?

   Gli account con un UID inferiore a 1000 generalmente sono account di sistema.

4. Come puoi ottenere l'elenco dei login attivi sul tuo sistema e anche il loro numero?

   Usando il comando `w`. Oltre a un elenco di tutti i login attivi, il comando mostra anche informazioni come il numero di utenti che hanno effettuato il login, il carico del sistema e il tempo di funzionamento.

5. Utilizzando il comando `grep`, abbiamo ottenuto le informazioni sull'utente `emma` riportate qui sotto:

   ```
   $ grep emma /etc/passwd
   emma:x:1000:1000:Emma Smith,42 Douglas St,555.555.5555,:/home/emma:/bin/ksh
   ```

   Completa gli spazi vuoti della tabella con le informazioni corrette utilizzando l'output del comando precedente.

Nome Utente	`emma`
Password	`x` - dovrebbe sempre essere `x` per un login utente valido e attivo
UID	`1000`
GID Primario	`1000`
GECOS	`Emma Smith,42 Douglas St,555.555.5555`
Directory Home	`/home/emma`
Shell	`/bin/ksh`

Risposte agli Esercizi Esplorativi

1. Confronta i risultati di `last` con quelli di `w` e `who`. Quali informazioni mancano a ciascuno dei comandi rispetto agli altri?

   Gli strumenti `w` e `who` elencano solo gli utenti attualmente connessi sul sistema, mentre `last` elenca anche gli utenti che si sono disconnessi. Il comando `w` mostra l'utilizzo del sistema, mentre `who` non lo fa.

2. Prova a eseguire i comandi `who` e `w -his`.
   - Quali informazioni sono state rimosse dall'output del comando `w` con le opzioni "no header" (`-h`) e "short" (`-s`)?

     Rispettivamente, non viene stampata l'intestazione (header), il che è utile per l'analisi, e non vengono riportate l'ora di accesso e le informazioni sulla CPU.

   - Quali informazioni sono state aggiunte all'output del comando `w` con l'opzione "ip address" (`-i`)?

     Questa opzione stampa l'indirizzo IP, invece di tentare la risoluzione DNS, stampando il nome host. Questa opzione di `w` ha una migliore corrispondenza con l'output predefinito del comando `last`.

3. Qual è il file che memorizza l'hash unidirezionale della password di un account utente?

   Il file `/etc/shadow` memorizza l'hash unidirezionale della password di un account utente poiché, a differenza di `/etc/passwd`, non è leggibile da un account utente ordinario e non privilegiato.

4. Quale file contiene l'elenco dei gruppi di cui è membro un account utente? Quale logica potrebbe essere utilizzata per stilarlo?

   Il file `/etc/group` ha un elenco CSV di nomi utente nell'ultimo campo ("members") di qualsiasi riga corrispondente a un gruppo.

   Qualsiasi riga nel file `/etc/group` in cui un utente è elencato nell'ultimo campo ("members") indica che quell'utente è un membro di quel gruppo, supponendo che sia formattato in modo corretto (CSV). Inoltre, il gruppo primario di un utente nel file `/etc/passwd` avrà anche una corrispondenza nel file `/etc/group` in termini di nome del gruppo e GID.

5. Per impostazione predefinita, uno o più (1+) dei seguenti file non sono leggibili da utenti ordinari senza privilegi. Quali sono?

- `/etc/group`
- `/etc/passwd`
- `/etc/shadow`
- `/etc/sudoers`

  I file `/etc/shadow` ed `/etc/sudoers` non sono leggibili per impostazione predefinita, tranne che dai servizi selezionati o dal superuser. Queste risposte dipendono dai sistemi e dai nomi utente utilizzati in laboratorio.

6. Come puoi cambiare, in modalità non interattiva, la shell di login dell'utente corrente nella Shell Korn (`/usr/bin/ksh`)?

```
$ chsh -s /usr/bin/ksh
```

7. Perché la directory home dell'utente `root` non si trova nella directory `/home`?

   Perché l'account `root` è necessario per individuare e correggere gli errori, che potrebbero includere problemi di file system relativi alla directory `/home`. In questi casi, `root` dovrebbe essere completamente funzionante anche quando il file system `/home` non è ancora disponibile.

5.2 Creazione di Utenti e Gruppi

Obiettivi LPI di riferimento

Linux Essentials version 1.6, Exam 010, Objective 5.2

Peso

2

Aree di Conoscenza Chiave

- Comandi gestione utenti e gruppi
- ID degli utenti

Di seguito è riportato un elenco parziale dei file, dei termini e dei comandi utilizzati

- `/etc/passwd`, `/etc/shadow`, `/etc/group`, `/etc/skel/`
- `useradd`, `groupadd`
- `passwd`

5.2 Lezione 1

Certificazione:	Linux Essentials
Versione:	1.6
Argomento:	5 Sicurezza e Permessi sui File
Obiettivo:	5.2 Creazione di Utenti e Gruppi
Lezione:	1 di 1

Introduzione

La gestione di utenti e gruppi su una macchina Linux è uno degli aspetti chiave dell'amministrazione di sistema: infatti Linux è un sistema operativo multiutente in cui più utenti possono utilizzare la stessa macchina contemporaneamente.

Le informazioni su utenti e gruppi sono memorizzate in quattro file all'interno dell'albero di directory `/etc/`:

`/etc/passwd`
: un file di sette campi delimitati da due punti contenente informazioni di base sugli utenti

`/etc/group`
: un file di quattro campi delimitati da due punti contenente informazioni di base sui gruppi

`/etc/shadow`
: un file di nove campi delimitati da due punti contenente le password utente crittografate

`/etc/gshadow`

un file di quattro campi delimitati da due punti contenente le password di gruppo crittografate

Tutti questi file vengono aggiornati da un insieme di strumenti a *Command Line* per la gestione di utenti e gruppi di cui parleremo più avanti in questa lezione. Possono anche essere gestiti da applicazioni grafiche, specifiche per ogni distribuzione Linux, che forniscono interfacce di gestione più semplici e immediate.

WARNING	Anche se i file sono file di testo normale, non vanno modificati direttamente. Per farlo usa sempre gli strumenti forniti con la tua distribuzione.

Il File `/etc/passwd`

`/etc/passwd` è un file leggibile da tutti che contiene un elenco di utenti, ciascuno su una riga separata:

```
frank:x:1001:1001::/home/frank:/bin/bash
```

Ogni riga è composta da sette campi delimitati da due punti:

Nome Utente

Il nome utilizzato quando l'utente effettua il login sul sistema.

Password

La password crittografata (o una `x` se vengono utilizzate le password shadow).

User ID (UID)

L'ID numerico assegnato all'utente nel sistema.

Group ID (GID)

Il numero del gruppo primario dell'utente nel sistema.

GECOS

Un campo di commento facoltativo, che viene utilizzato per aggiungere ulteriori informazioni sull'utente (come per esempio il nome completo). Il campo può contenere più voci separate da virgole.

Directory Home

Il percorso assoluto della directory home dell'utente.

Shell

Il percorso assoluto del programma che viene avviato automaticamente quando l'utente effettua il login sul sistema (di solito una shell interattiva come `/bin/bash`).

Il File `/etc/group`

`/etc/group` è un file leggibile da tutti che contiene un elenco di gruppi, ciascuno su una riga separata:

```
developer:x:1002:
```

Ogni riga è composta da quattro campi delimitati da due punti:

Nome del gruppo

Il nome del gruppo.

Password del gruppo

La password crittografata del gruppo (o una `x` se vengono utilizzate le password shadow).

Group ID (GID)

L'ID numerico assegnato al gruppo nel sistema.

Lista dei membri

Un elenco, delimitato da virgole, di utenti appartenenti al gruppo, a eccezione di quelli per i quali questo è il gruppo primario.

Il File `/etc/shadow`

`/etc/shadow` è un file leggibile solo da *root* e dagli utenti con privilegi di root e contiene le password crittografate degli utenti, ciascuna su una riga separata:

```
frank:$6$i9gjM4Md4MuelZCd$7jJa8Cd2bbADFH4dwtfvTvJLOYCCCBf/.jYbK1IMYx7Wh4fErXcc2xQVU2N1gb97yI
YaiqH.jjJammzof2Jfr/:18029:0:99999:7:::
```

Ogni riga è composta da nove campi delimitati da due punti

Nome Utente

Il nome utilizzato quando l'utente effettua il login sul sistema.

Password crittografata

La password crittografata dell'utente (se il valore è `!`, l'account è bloccato).

Data dell'ultima modifica della password

La data dell'ultima modifica della password, come numero di giorni dal 01/01/1970. Un valore pari a `0` significa che l'utente deve cambiare la password al prossimo accesso.

Durata minima della password

Il numero minimo di giorni, dopo una modifica della password, che deve trascorrere prima che all'utente sia consentito modificare nuovamente la password.

Durata massima della password

Il numero massimo di giorni che deve trascorrere prima che sia richiesta una modifica della password.

Periodo di avviso della password

Il numero di giorni, prima della scadenza della password, durante i quali l'utente viene avvisato che la password deve essere modificata.

Periodo di inattività della password

Il numero di giorni, dopo la scadenza di una password, durante i quali l'utente deve aggiornare la password. Trascorso questo periodo, se l'utente non modifica la password, l'account verrà disabilitato.

Data di scadenza dell'account

La data, come numero di giorni dal 01/01/1970, in cui l'account utente verrà disabilitato. Un campo vuoto significa che l'account utente non scadrà mai.

Campo riservato

Un campo riservato per usi futuri.

Il File `/etc/gshadow`

`/etc/gshadow` è un file leggibile solo da *root* e dagli utenti con privilegi di root e contiene le password crittografate dei gruppi, ciascuna su una riga separata:

```
developer:$6$7QUIhUX1WdO6$H7kOYgsboLkDseFHpk04lwAtweSUQHipoxIgo83QNDxYtYwgmZTCU0qSCuCkErmyR2
63rvHiLctZVDR7Ya9Ai1::
```

Ogni riga è composta da quattro campi delimitati da due punti:

Nome del gruppo

Il nome del gruppo.

Password crittografata

La password crittografata del gruppo (viene utilizzata quando un utente, che non è un membro del gruppo, desidera unirsi al gruppo utilizzando il comando `newgrp` — se la password inizia con `!` a nessuno è consentito accedere al gruppo con `newgrp`).

Amministratori del gruppo

Un elenco delimitato da virgole degli amministratori del gruppo (possono cambiare la password del gruppo e possono aggiungere o rimuovere membri del gruppo con il comando `gpasswd`).

Membri del gruppo

Un elenco delimitato da virgole dei membri del gruppo.

Ora che abbiamo visto dove vengono archiviate le informazioni su utenti e gruppi parliamo degli strumenti più importanti della Command Line per aggiornare questi file.

Aggiungere ed Eliminare Account Utenti

In Linux è possibile aggiungere un nuovo account utente con il comando `useradd` ed eliminarlo con il comando `userdel`.

Se desideri creare un nuovo account utente chiamato `frank` con impostazioni predefinite, puoi eseguire quanto segue:

```
# useradd frank
```

Dopo aver creato il nuovo utente puoi impostare una password usando `passwd`:

```
# passwd frank
Changing password for user frank.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

Entrambi questi comandi richiedono privilegi di root. Quando esegui il comando `useradd`, le informazioni sull'utente e sul gruppo memorizzate nei database delle password e dei gruppi vengono aggiornate per l'account utente appena creato e, se specificato, viene creata la directory

home del nuovo utente e un gruppo con lo stesso nome dell'account utente.

TIP

Ricorda che puoi sempre utilizzare l'utility `grep` per filtrare i database delle password e dei gruppi, così da visualizzare solo la voce che fa riferimento a uno specifico utente o gruppo. Per l'esempio sopra riportato puoi usare:

```
cat /etc/passwd | grep frank
```

o

```
grep frank /etc/passwd
```

per visualizzare le informazioni di base sull'account `frank` appena creato.

Le opzioni più importanti del comando `useradd` sono:

`-c`
Crea un nuovo account utente con commenti personalizzati (per esempio il nome completo).

`-d`
Crea un nuovo account utente con una directory home personalizzata.

`-e`
Crea un nuovo account utente impostando una data specifica in cui verrà disabilitato.

`-f`
Crea un nuovo account utente impostando il numero di giorni, dopo la scadenza della password, durante i quali l'utente deve aggiornare la password stessa.

`-g`
Crea un nuovo account utente con uno specifico GID.

`-G`
Crea un nuovo account utente aggiungendolo a più gruppi secondari.

`-m`
Crea un nuovo account utente con la sua directory home.

`-M`
Crea un nuovo account utente senza la sua directory home.

`-s`

Crea un nuovo account utente con una specifica shell di login.

`-u`

Crea un nuovo account utente con uno specifico UID.

Una volta creato il nuovo account utente puoi usare i comandi `id` e `groups` per scoprire il suo UID, il suo GID e i gruppi a cui appartiene.

```
# id frank
uid=1000(frank) gid=1000(frank) groups=1000(frank)
# groups frank
frank : frank
```

TIP

Ricordati di controllare ed eventualmente modificare il file `/etc/login.defs` che definisce i parametri di configurazione che controllano la creazione di utenti e gruppi. Per esempio, puoi impostare l'intervallo di UID e GID che possono essere assegnati ai nuovi account utente e di gruppo, puoi specificare che non è necessario utilizzare l'opzione `-m` per creare la directory home del nuovo utente e specificare se il sistema deve creare automaticamente un nuovo gruppo per ogni nuovo utente.

Per eliminare un account utente puoi usare il comando `userdel`. In particolare, questo comando aggiorna le informazioni memorizzate nei database degli account, eliminando tutte le voci che fanno riferimentio all'utente specificato. L'opzione `-r` rimuove anche la directory home dell'utente e tutto il suo contenuto, insieme allo spool di posta dell'utente. Gli altri file, che si trovano altrove, devono essere cercati ed eliminati manualmente.

```
# userdel -r frank
```

Anche in questo caso sono necessari i privilegi di root per eliminare gli account utente.

La Directory Scheletro

Quando aggiungi un nuovo account utente, anche creando la sua directory home, la directory home appena creata viene popolata con file e cartelle che vengono copiate dalla directory scheletro (*skeleton directory* - per impostazione predefinita `/etc/skel`). L'idea alla base di tutto ciò è semplice: un amministratore di sistema desidera aggiungere nuovi utenti con gli stessi file e le stesse directory nella propria home. Pertanto, se si desidera personalizzare i file e le cartelle che vengono creati automaticamente nella directory home dei nuovi account utente, è necessario

aggiungere questi nuovi file e cartelle alla directory scheletro.

TIP Nota che i file di profilo che di solito si trovano nella directory scheletro sono file nascosti. Pertanto, se vuoi elencare tutti i file e le cartelle nella directory scheletro, che verranno copiati nella directory home degli utenti appena creati, devi usare il comando `ls -Al`.

Aggiungere ed Eliminare Gruppi

Per quanto riguarda la gestione dei gruppi, puoi aggiungere o eliminare gruppi usando i comandi `groupadd` e `groupdel`.

Se vuoi creare un nuovo gruppo chiamato `developer`, puoi eseguire il seguente comando come root:

```
# groupadd -g 1090 developer
```

L'opzione `-g` di questo comando crea un gruppo con uno specifico GID.

Se vuoi eliminare il gruppo `developer`, puoi eseguire quanto segue:

```
# groupdel developer
```

WARNING Ricorda che quando aggiungi un nuovo account utente, il gruppo primario e i gruppi secondari a cui appartiene devono esistere prima di eseguire il comando `useradd`. Inoltre, non è possibile eliminare un gruppo se è il gruppo primario di un account utente.

Il Comando `passwd`

Questo comando viene utilizzato principalmente per modificare la password di un utente. Ogni utente può modificare la propria password, ma solo root può modificare la password di qualsiasi utente.

A seconda dell'opzione di `passwd` usata, puoi controllare specifici aspetti relativi all'invecchiamento della password:

`-d`
: Elimina la password di un account utente (impostando così una password vuota, rendendolo un account senza password).

`-e`

Forza l'account utente a modificare la password.

`-l`

Blocca l'account utente (la password crittografata viene preceduta da un punto esclamativo).

`-u`

Sblocca l'account utente (rimuove il punto esclamativo).

`-S`

Mostra informazioni sullo stato della password di uno specifico account.

Queste opzioni sono disponibili solo per root. Per vedere l'elenco completo delle opzioni, fai riferimento alle pagine *man*.

Esercizi Guidati

1. Indica a quale file si riferisce ciascuna delle seguenti voci:
   - `developer:x:1010:frank,grace,dave`
   - `root:x:0:0:root:/root:/bin/bash`
   - `henry:$1$.AbCdEfGh123456789A1b2C3d4.:18015:20:90:5:30::`
   - `henry:x:1000:1000:User Henry:/home/henry:/bin/bash`
   - `staff:!:dave:carol,emma`

2. Osserva il seguente output e rispondi alle prossime sette domande:

```
# cat /etc/passwd | tail -3
dave:x:1050:1050:User Dave:/home/dave:/bin/bash
carol:x:1051:1015:User Carol:/home/carol:/bin/sh
henry:x:1052:1005:User Henry:/home/henry:/bin/tcsh
# cat /etc/group | tail -3
web_admin:x:1005:frank,emma
web_developer:x:1010:grace,kevin,christian
dave:x:1050:
# cat /etc/shadow | tail -3
dave:$6$AbCdEfGh123456789A1b2C3D4e5F6G7h8i9:0:20:90:7:30::
carol:$6$q1w2e3r4t5y6u7i8AbcDeFgHiLmNoPqRsTu:18015:0:60:7:::
henry:!$6$123456789aBcDeFgHa1B2c3d4E5f6g7H8I9:18015:0:20:5:::
# cat /etc/gshadow | tail -3
web_admin:!:frank:frank,emma
web_developer:!:kevin:grace,kevin,christian
dave:!::
```

   - Qual è l'ID utente (UID) e l'ID di gruppo (GID) di `carol`?
   - Quale shell è impostata per `dave` e per `henry`?

- Qual è il nome del gruppo primario di `henry`?

- Quali sono i membri del gruppo `web_developer`? Quali di questi sono amministratori del gruppo?

- Quale utente non può effettuare il login sul sistema?

- Quale utente deve cambiare la password la prossima volta che effettuerà il login sul sistema?

- Quanti giorni devono passare prima che venga richiesta una modifica della password per `carol`?

Esercizi Esplorativi

1. Lavorando come root, esegui il comando `useradd -m dave` per aggiungere un nuovo account utente. Quali operazioni esegue questo comando? Supponi che `CREATE_HOME` e `USERGROUPS_ENAB` in `/etc/login.defs` siano impostati su *yes*.

2. Ora che hai creato l'account `dave`, questo utente può effettuare il login sul sistema?

3. Identifica l'ID utente (UID) e l'ID di gruppo (GID) di `dave` e tutti i membri del gruppo `dave`.

4. Crea i gruppi `sys_admin`, `web_admin` e `db_admin` e individua i loro ID di gruppo (GID).

5. Aggiungi un nuovo account utente chiamato `carol` con UID 1035 e imposta `sys_admin` come suo gruppo principale e `web_admin` e `db_admin` come suoi gruppi secondari.

6. Elimina gli account utente `dave` e `carol` e i gruppi `sys_admin`, `web_admin` e `db_admin` che hai precedentemente creato.

7. Esegui il comando `ls -l /etc/passwd /etc/group /etc/shadow /etc/gshadow` e descrivi l'output ottenuto in termini di permessi sui file. Quali di questi quattro file sono oscurati per motivi di sicurezza? Supponi che il tuo sistema utilizzi le password shadow.

8. Esegui il comando `ls -l /usr/bin/passwd`. Quale bit speciale è impostato e qual è il suo significato?

Sommario

In questa lezione hai imparato:

- I fondamenti della gestione di utenti e gruppi in Linux;
- Come gestire le informazioni di utenti e gruppi archiviate nei database delle password e dei gruppi;
- Come gestire la directory scheletro;
- Come aggiungere e rimuovere account utenti;
- Come aggiungere e rimuovere account di gruppi;
- Come modificare la password di un account utente.

In questa lezione sono stati discussi i seguenti comandi:

`useradd`
: Crea un nuovo account utente.

`groupadd`
: Crea un nuovo account di gruppo.

`userdel`
: Elimina un account utente.

`groupdel`
: Elimina un account di gruppo.

`passwd`
: Modifica la password di un account utente e controlla tutti gli aspetti relativi all'invecchiamento della password.

Risposte agli Esercizi Guidati

1. Indica a quale file si riferisce ciascuna delle seguenti voci:
   - `developer:x:1010:frank,grace,dave`

     `/etc/group`
   - `root:x:0:0:root:/root:/bin/bash`

     `/etc/passwd`
   - `henry:$1$.AbCdEfGh123456789A1b2C3d4.:18015:20:90:5:30::`

     `/etc/shadow`
   - `henry:x:1000:1000:User Henry:/home/henry:/bin/bash`

     `/etc/passwd`
   - `staff:!:dave:carol,emma`

     `/etc/gshadow`
2. Osserva il seguente output e rispondi alle prossime sette domande:

```
# cat /etc/passwd | tail -3
dave:x:1050:1050:User Dave:/home/dave:/bin/bash
carol:x:1051:1015:User Carol:/home/carol:/bin/sh
henry:x:1052:1005:User Henry:/home/henry:/bin/tcsh
# cat /etc/group | tail -3
web_admin:x:1005:frank,emma
web_developer:x:1010:grace,kevin,christian
dave:x:1050:
# cat /etc/shadow | tail -3
dave:$6$AbCdEfGh123456789A1b2C3D4e5F6G7h8i9:0:20:90:7:30::
carol:$6$q1w2e3r4t5y6u7i8AbcDeFgHiLmNoPqRsTu:18015:0:60:7:::
henry:!$6$123456789aBcDeFgHa1B2c3d4E5f6g7H8I9:18015:0:20:5:::
# cat /etc/gshadow | tail -3
web_admin:!:frank:frank,emma
web_developer:!:kevin:grace,kevin,christian
dave:!::
```

   - Qual è l'ID utente (UID) e l'ID di gruppo (GID) di `carol`?

Lo UID è 1051 e il GID è 1015 (il terzo e il quarto campo in `/etc/passwd`).

- Quale shell è impostata per `dave` e per `henry`?

  `dave` usa /bin/bash, mentre `henry` usa /bin/tcsh (il settimo campo in `/etc/passwd`).

- Qual è il nome del gruppo primario di `henry`?

  Il nome del gruppo è `web_admin` (il primo campo in `/etc/group`).

- Quali sono i membri del gruppo `web_developer`? Quali di questi sono amministratori del gruppo?

  I membri sono `grace`, `kevin` e `christian` (il quarto campo in `/etc/group`), ma solo `kevin` è l'amministratore del gruppo (il terzo campo in `/etc/gshadow`) .

- Quale utente non può effettuare il login sul sistema?

  L'account utente `henry` è bloccato (ha un punto esclamativo davanti agli hash della password in `/etc/shadow`).

- Quale utente deve cambiare la password la prossima volta che effettuerà il login sul sistema?

  Se il terzo campo (Data dell'ultima modifica della password) in `/etc/shadow` è 0, l'utente deve cambiare la propria password la prossima volta che effettuerà il login sul sistema. Pertanto, `dave` deve cambiare la sua password.

- Quanti giorni devono passare prima che venga richiesta una modifica della password per `carol`?

  60 giorni (il quinto campo in `/etc/shadow`).

Risposte agli Esercizi Esplorativi

1. Lavorando come root, esegui il comando `useradd -m dave` per aggiungere un nuovo account utente. Quali operazioni esegue questo comando? Supponi che `CREATE_HOME` e `USERGROUPS_ENAB` in `/etc/login.defs` siano impostati su *yes*.

   Il comando aggiunge un nuovo utente, chiamato `dave`, all'elenco degli utenti del sistema. Viene creata la directory home di `dave` (per impostazione predefinita `/home/dave`) e i file e le directory contenute nella directory scheletro (skeleton directory) vengono copiati nella directory home. Infine, viene creato un nuovo gruppo con lo stesso nome dell'account utente.

2. Ora che hai creato l'account `dave`, questo utente può effettuare il login sul sistema?

   No, poiché l'account `dave` è bloccato (nota il punto esclamativo in `/etc/shadow`).

   ```
   # cat /etc/shadow | grep dave
   dave:!:18015:0:99999:7:::
   ```

   Se imposti una password per `dave`, l'account verrà sbloccato. Puoi farlo usando il comando `passwd`.

   ```
   # passwd dave
   Changing password for user dave.
   New UNIX password:
   Retype new UNIX password:
   passwd: all authentication tokens updated successfully.
   ```

3. Identifica l'ID utente (UID) e l'ID di gruppo (GID) di `dave` e tutti i membri del gruppo `dave`.

   ```
   # cat /etc/passwd | grep dave
   dave:x:1015:1019::/home/dave:/bin/sh
   # cat /etc/group | grep 1019
   dave:x:1019:
   ```

   Lo UID e il GID di `dave` sono rispettivamente 1015 e 1019 (il terzo e il quarto campo in `/etc/passwd`), e il gruppo `dave` non ha alcun membro (il quarto campo in `/etc/group` è vuoto) .

4. Crea i gruppi `sys_admin`, `web_admin` e `db_admin` e individua i loro ID di gruppo (GID).

```
# groupadd sys_admin
# groupadd web_admin
# groupadd db_admin
# cat /etc/group | grep admin
sys_admin:x:1020:
web_admin:x:1021:
db_admin:x:1022:
```

I GID dei gruppi `sys_admin`, `web_admin` e `db_admin` sono rispettivamente 1020, 1021 e 1022.

5. Aggiungi un nuovo account utente chiamato `carol` con UID 1035 e imposta `sys_admin` come suo gruppo principale e `web_admin` e `db_admin` come suoi gruppi secondari.

```
# useradd -u 1035 -g 1020 -G web_admin,db_admin carol
# id carol
uid=1035(carol) gid=1020(sys_admin) groups=1020(sys_admin),1021(web_admin),1022(db_admin)
```

6. Elimina gli account utente `dave` e `carol` e i gruppi `sys_admin`, `web_admin` e `db_admin` che hai precedentemente creato.

```
# userdel -r dave
# userdel -r carol
# groupdel sys_admin
# groupdel web_admin
# groupdel db_admin
```

7. Esegui il comando `ls -l /etc/passwd /etc/group /etc/shadow /etc/gshadow` e descrivi l'output ottenuto in termini di permessi sui file. Quali di questi quattro file sono oscurati per motivi di sicurezza? Supponi che il tuo sistema utilizzi le password shadow.

```
# ls -l /etc/passwd /etc/group /etc/shadow /etc/gshadow
-rw-r--r-- 1 root root    853 mag  1 08:00 /etc/group
-rw-r----- 1 root shadow 1203 mag  1 08:00 /etc/gshadow
-rw-r--r-- 1 root root   1354 mag  1 08:00 /etc/passwd
-rw-r----- 1 root shadow 1563 mag  1 08:00 /etc/shadow
```

I file `/etc/passwd` ed `/etc/group` sono leggibili da chiunque e sono oscurati per motivi di sicurezza. Quando vengono utilizzate le password shadow, puoi notare una `x` nel secondo campo di questi file poichè le password crittografate di utenti e gruppi sono memorizzate in

`/etc/shadow` ed `/etc/gshadow`, che sono leggibili solo da root e, in alcuni sistemi, anche dai membri appartenenti al gruppo `shadow`.

8. Esegui il comando `ls -l /usr/bin/passwd`. Quale bit speciale è impostato e qual è il suo significato?

```
# ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 42096 mag 17  2015 /usr/bin/passwd
```

Il comando `passwd` ha impostato il bit SUID (il quarto carattere della riga), il che significa che il comando viene eseguito con i privilegi del proprietario del file (quindi root). Questo è il motivo per cui gli utenti ordinari possono modificare la propria password.

5.3 Gestione delle Autorizzazioni e delle Proprietà dei File

Obiettivi LPI di riferimento

Linux Essentials version 1.6, Exam 010, Objective 5.3

Peso

2

Aree di Conoscenza Chiave

- Autorizzazioni e proprietà di file e directory

Di seguito è riportato un elenco parziale dei file, dei termini e dei comandi utilizzati

- `ls -l`, `ls -a`
- `chmod`, `chown`

5.3 Lezione 1

Certificazione:	Linux Essentials
Versione:	1.6
Argomento:	5 Sicurezza e Permessi sui File
Obiettivo:	5.3 Gestione delle Autorizzazioni e delle Proprietà dei File
Lezione:	1 di 1

Introduzione

Essendo un sistema multiutente, Linux ha bisogno di un modo per tenere traccia di chi sia il proprietario di ciascun file e se un utente sia autorizzato o meno a eseguire azioni su quel file. Questo per garantire la privacy degli utenti che potrebbero voler mantenere riservato il contenuto dei propri file, nonché per garantire la collaborazione rendendo alcuni file accessibili a più utenti.

Ciò avviene attraverso un sistema di permessi a tre livelli: ogni file sul disco è di proprietà di un utente e di un gruppo di utenti e dispone di tre tipi di permessi: uno per il suo proprietario, uno per il gruppo che possiede il file e uno per tutti gli altri. In questa lezione imparerai come individuare i permessi di un file e come modificarli.

Ricercare Informazioni su File e Directory

Il comando `ls` viene utilizzato per ottenere un elenco dei contenuti di qualsiasi directory. In questa forma base, ciò che ottieni sono solo i nomi dei file:

```
$ ls
```

```
Another_Directory  picture.jpg  text.txt
```

Ma ci sono molte più informazioni disponibili per ogni file, inclusi il tipo, la dimensione, chi lo possiede (*ownership*) e altro. Per vedere queste informazioni devi chiedere a `ls` un elenco di "forma lunga" (*long form*), utilizzando il parametro `-l`:

```
$ ls -l
total 536
drwxrwxr-x 2 carol carol   4096 Dec 10 15:57 Another_Directory
-rw------- 1 carol carol 539663 Dec 10 10:43 picture.jpg
-rw-rw-r-- 1 carol carol   1881 Dec 10 15:57 text.txt
```

Ogni colonna dell'output sopra riportato ha un significato specifico:

- La *prima* colonna dell'elenco mostra il tipo di file e i permessi.

  Per esempio, in `drwxrwxr-x`:

  - Il primo carattere, `d`, indica il tipo di file.
  - I successivi tre caratteri, `rwx`, indicano i permessi del proprietario del file, indicato anche come *utente* (*user*) o `u`.
  - I successivi tre caratteri, `rwx`, indicano i permessi del *gruppo* (*group*) che possiede il file, indicato anche come `g`.
  - Gli ultimi tre caratteri, `r-x`, indicano i permessi per tutti gli altri, noti anche come *others* o `o`.
- La *seconda* colonna indica il numero di hard link che puntano a quel file. Per una directory è il numero di sottodirectory, più un link a se stessa (`.`) e alla directory padre (`..`).
- La *terza* e la *quarta* colonna mostrano informazioni di ownership: rispettivamente l'utente e il gruppo che possiede il file.
- La *quinta* colonna mostra la dimensione del file, in byte.
- La *sesta* colonna mostra la data e l'ora precise, o *timestamp*, dell'ultima modifica del file.
- La *settima* e ultima colonna mostra il nome del file.

Se desideri vedere le dimensioni dei file in un formato "leggibile dall'uomo", aggiungi il parametro `-h` a `ls`. I file di dimensioni inferiori a 1 kilobyte avranno la dimensione mostrata in byte. I file di dimensioni maggiori di 1 kilobyte e inferiori a 1 megabyte avranno una `K` dopo la dimensione, a indicare che la dimensione è in kilobyte. Lo stesso vale per le dimensioni dei file negli intervalli dei megabyte (`M`) e dei gigabyte (`G`):

```
$ ls -lh
total 1,2G
drwxrwxr-x 2 carol carol 4,0K Dec 10 17:59 Another_Directory
----r--r-- 1 carol carol    0 Dec 11 10:55 foo.bar
-rw-rw-r-- 1 carol carol 1,2G Dec 20 18:22 HugeFile.zip
-rw------- 1 carol carol 528K Dec 10 10:43 picture.jpg
---xr-xr-x 1 carol carol   33 Dec 11 10:36 test.sh
-rwxr--r-- 1 carol carol 1,9K Dec 20 18:13 text.txt
-rw-rw-r-- 1 carol carol 2,6M Dec 11 22:14 Zipped.zip
```

Per mostrare solo le informazioni su un insieme specifico di file, aggiungi i nomi di questi file a `ls`:

```
$ ls -lh HugeFile.zip test.sh
total 1,2G
-rw-rw-r-- 1 carol carol 1,2G Dec 20 18:22 HugeFile.zip
---xr-xr-x 1 carol carol   33 Dec 11 10:36 test.sh
```

E per le Directory?

Se provi a ricercare informazioni su una directory usando `ls -l`, il comando ti mostrerà invece un elenco dei contenuti della directory:

```
$ ls -l Another_Directory/
total 0
-rw-r--r-- 1 carol carol 0 Dec 10 17:59 another_file.txt
```

Per evitare questo comportamento e richiedere informazioni sulla directory stessa, aggiungi l'opzione `-d` a `ls`:

```
$ ls -l -d Another_Directory/
drwxrwxr-x 2 carol carol 4096 Dec 10 17:59 Another_Directory/
```

Visualizzare File Nascosti

L'elenco che abbiamo recuperato utilizzando `ls -l` visto in precedenza è incompleto:

```
$ ls -l
```

```
total 544
drwxrwxr-x 2 carol carol   4096 Dec 10 17:59 Another_Directory
-rw------- 1 carol carol 539663 Dec 10 10:43 picture.jpg
-rw-rw-r-- 1 carol carol   1881 Dec 10 15:57 text.txt
```

Ci sono altri tre file in questa directory, ma sono nascosti. In Linux, i file il cui nome inizia con un punto (`.`) vengono automaticamente nascosti. Per vederli dobbiamo aggiungere l'opzione `-a` a `ls`:

```
$ ls -l -a
total 544
drwxrwxr-x 3 carol carol   4096 Dec 10 16:01 .
drwxrwxr-x 4 carol carol   4096 Dec 10 15:56 ..
drwxrwxr-x 2 carol carol   4096 Dec 10 17:59 Another_Directory
-rw------- 1 carol carol 539663 Dec 10 10:43 picture.jpg
-rw-rw-r-- 1 carol carol   1881 Dec 10 15:57 text.txt
-rw-r--r-- 1 carol carol      0 Dec 10 16:01 .thisIsHidden
```

Il file `.thisIsHidden` è semplicemente nascosto poiché il suo nome inizia con un `.`.

Le directory `.` e `..` tuttavia sono directory speciali: `.` è un puntatore alla directory corrente, mentre `..` è un puntatore alla directory padre (la directory che contiene la directory corrente). In Linux, ogni directory contiene almeno queste due directory speciali.

TIP Puoi combinare più opzioni per `ls` (e molti altri comandi Linux). Per esempio, `ls -l -a` può essere scritto come `ls -la`.

Comprendere i Tipi di File

Abbiamo già detto che la prima lettera nell'output di `ls -l` descrive il tipo di file. I tre tipi di file più comuni sono:

`-` (file normale)

Un file può contenere dati di qualsiasi tipo. I file possono essere modificati, spostati, copiati ed eliminati.

`d` (directory)

Una directory contiene altri file o directory e aiuta a organizzare il *file system*. Tecnicamente, le directory sono un tipo speciale di file.

`l` (soft link)

Questo "file" è un puntatore a un altro file o a una directory che si trova altrove nel filesystem.

Oltre a questi, ci sono almeno altri tre tipi di file che dovresti conoscere, ma che non rientrano nell'ambito di questa lezione:

`b` (dispositivo a blocchi)

Questo file rappresenta un dispositivo virtuale o fisico, solitamente dei dischi o altri tipi di dispositivi di archiviazione. Per esempio, il primo disco rigido nel sistema potrebbe essere rappresentato da `/dev/sda`.

`c` (dispositivo a caratteri)

Questo file rappresenta un dispositivo virtuale o fisico. I terminali (come il terminale principale su `/dev/ttyS0`) e le porte seriali sono esempi comuni di dispositivi a caratteri.

`s` (socket)

I *socket* fungono da "condotti" per il passaggio di informazioni tra due programmi.

WARNING	Non modificare nessun permesso su dispositivi a blocchi, dispositivi a caratteri o socket, a meno che tu non sappia cosa stai facendo. Questo potrebbe impedire al tuo sistema di funzionare!

Comprendere i Permessi

Nell'output di `ls -l` i permessi di un file sono mostrati subito dopo il tipo di file, come tre gruppi di tre caratteri ciascuno, secondo l'ordine `r`, `w` e `x`. Vedremo ora cosa significano. Tieni presente che un trattino `-` rappresenta la mancanza di un certo permesso.

Permessi sui File

`r`

Sta per *read* (lettura) e ha un *valore ottale* pari a `4` (non preoccuparti, discuteremo a breve di valori ottali). Ciò significa il permesso di aprire un file e leggerne il contenuto.

`w`

Sta per *write* (scrittura) e ha un valore ottale pari a `2`. Ciò significa il permesso di modificare o eliminare un file.

`x`

Sta per *execute* (esecuzione) e ha un valore ottale pari a `1`. Ciò significa che il file può essere eseguito come un eseguibile o uno script.

Quindi, per esempio, un file con permessi pari a `rw-` può essere letto e scritto, ma non può essere eseguito.

Permessi sulle Directory

`r`

Sta per *read* (lettura) e ha un valore ottale pari a `4`. Ciò significa il permesso di leggere il contenuto della directory, come i nomi dei file, ma *non* implica il permesso di leggere i file stessi.

`w`

Sta per *write* (scrittura) e ha un valore ottale pari a 2. Ciò significa il permesso di creare o eliminare file in una directory o di modificarne i nomi, i permessi e i proprietari. Se un utente dispone dei permessi di scrittura su una directory, può modificare i permessi di qualsiasi file nella directory, anche se l'utente non ha permessi sul file o se il file è di proprietà di un altro utente.

`x`

Sta per *execute* e ha un valore ottale pari a `1`. Ciò significa il permesso di entrare in una directory, ma non di elencare i suoi file (per questo, è necessario il permesso `r`).

L'ultimo bit per le directory può sembrare un po' confusionario. Immaginiamo, per esempio, di avere una directory chiamata `Another_Directory` con i seguenti permessi:

```
$ ls -ld Another_Directory/
d--xr-xr-x 2 carol carol 4,0K Dec 20 18:46 Another_Directory
```

Immagina anche che all'interno di questa directory tu abbia uno script di shell chiamato `hello.sh` con i seguenti permessi:

```
-rwxr-xr-x 1 carol carol 33 Dec 20 18:46 hello.sh
```

Se sei l'utente `carol` e cerchi di elencare il contenuto di `Another_Directory`, riceverai un messaggio di errore, poiché il tuo utente non ha i permessi di lettura per quella directory:

```
$ ls -l Another_Directory/
ls: cannot open directory 'Another_Directory/': Permission denied
```

Tuttavia, l'utente `carol` *dispone* dei permessi di esecuzione, il che significa che può entrare nella directory. Pertanto, l'utente `carol` può accedere ai file all'interno della directory, purché abbia i permessi corretti *per i rispettivi file*. In questo esempio, l'utente ha i permessi completi per lo script `hello.sh`; quindi *può* eseguire lo script anche se *non può* leggere il contenuto della

directory che lo contiene. Tutto ciò di cui ha bisogno è il nome del file completo.

```
$ sh Another_Directory/hello.sh
Hello LPI World!
```

Come detto in precedenza i permessi sono specificati in sequenza: prima per il proprietario del file, poi per il gruppo proprietario e infine per gli altri utenti. Ogni volta che qualcuno tenta di eseguire un'azione sul file, i permessi vengono verificati nello stesso ordine. Per prima cosa il sistema controlla se l'utente corrente possiede il file e, se questo è vero, applica solo la prima serie di permessi. In caso contrario, controlla se l'utente corrente appartiene al gruppo proprietario del file. In tal caso, applica solo la seconda serie di permessi. In ogni altro caso, il sistema applica la terza serie di permessi. Ciò significa che se l'utente corrente è il proprietario del file, sono applicati solo i permessi del proprietario, anche se i permessi di gruppo o degli altri fossero superiori a quelli del proprietario.

Modificare i Permessi dei File

Il comando `chmod` viene utilizzato per modificare i permessi di un file e necessita almeno di due parametri: il primo descrive quali permessi devono essere modificati, mentre il secondo punta al file o alla directory su cui verrà effettuata la modifica. Tuttavia, i permessi per la modifica possono essere indicati in due modi diversi, o "modalità".

La prima, chiamata *modalità simbolica*, offre un controllo *granulare*, consentendo di aggiungere o revocare un singolo permesso senza modificare gli altri all'interno del set di permessi. L'altra, chiamata *modalità numerica*, è più facile da ricordare e più veloce da usare se desideri impostare tutti i valori dei permessi contemporaneamente.

Entrambe le modalità porteranno allo stesso risultato finale. Quindi, per esempio, i comandi:

```
$ chmod ug+rw-x,o-rwx text.txt
```

e

```
$ chmod 660 text.txt
```

produrranno esattamente lo stesso output, ovvero un file con impostati i seguenti permessi:

```
-rw-rw---- 1 carol carol  765 Dec 20 21:25 text.txt
```

Capiamo ora come funziona ciascuna modalità.

Modalità Simbolica

Quando si indicano quali permessi cambiare in *modalità simbolica*, il(i) primo(i) carattere(i) indica(no) quali permessi saranno oggetto della modifica: quelli per l'utente (`u`), per il gruppo (`g`), per gli altri (`o`), e/o per tutti e tre insieme (`a`).

A questo punto devi dire al comando che cosa fare: puoi concedere un permesso (`+`), revocare un permesso (`-`) o impostare un permesso a un valore specifico (`=`).

Infine, devi specificare quale permesso si desidera impostare: lettura (`r`), scrittura (`w`) o esecuzione (`x`).

Per esempio, immagina di avere un file chiamato `text.txt` con i seguenti permessi:

```
$ ls -l text.txt
-rw-r--r-- 1 carol carol 765 Dec 20 21:25 text.txt
```

Se desideri concedere i permessi di scrittura ai membri del gruppo che possiede il file, devi utilizzare il parametro `g+w`. È più facile se pensi in questo modo: "Per il gruppo (`g`), concedi (`+`) i permessi di scrittura (`w`)". Quindi, il comando da usare è:

```
$ chmod g+w text.txt
```

Controlliamo il risultato con `ls`:

```
$ ls -l text.txt
-rw-rw-r-- 1 carol carol 765 Dec 20 21:25 text.txt
```

Se vuoi rimuovere i permessi di lettura per il proprietario dello stesso file, puoi pensare in questo modo: "Per l'utente (`u`) revoca (`-`) i permessi di lettura (`r`)". Quindi il parametro è `u-r` come puoi vedere qui sotto:

```
$ chmod u-r text.txt
$ ls -l text.txt
--w-rw-r-- 1 carol carol 765 Dec 20 21:25 text.txt
```

E se volessimo impostare i permessi esattamente a `rw-` per tutti? Pensalo in questo modo: "Per

tutti (`a`), imposta esattamente (`=`) i permessi di lettura (`r`), scrittura (`w`) ma non di esecuzione (`-`)":

```
$ chmod a=rw- text.txt
$ ls -l text.txt
-rw-rw-rw- 1 carol carol 765 Dec 20 21:25 text.txt
```

Ovviamente è possibile modificare più permessi contemporaneamente. In questo caso, separali con una virgola (`,`):

```
$ chmod u+rwx,g-x text.txt
$ ls -lh text.txt
-rwxrw-rw- 1 carol carol 765 Dec 20 21:25 text.txt
```

L'esempio sopra riportato può essere letto come: "Per l'utente (`u`) concedi (`+`) i permessi di lettura, scrittura ed esecuzione (`rwx`), per il gruppo (`g`) revoca (`-`) i permessi di esecuzione (`x`)".

Quando viene eseguito su una directory, `chmod` modifica solo i permessi della directory. `chmod` ha una modalità ricorsiva, utile quando vuoi cambiare i permessi per "tutti i file all'interno di una directory e delle sue sottodirectory". Per usarla, aggiungi l'opzione `-R` dopo il nome del comando e prima dei permessi da modificare, in questo modo:

```
$ chmod -R u+rwx Another_Directory/
```

Questo comando può essere letto nel seguente modo: "Ricorsivamente (`-R`), per l'utente (`u`) concedi (`+`) permessi di lettura, scrittura ed esecuzione (`rwx`)".

WARNING Fai attenzione e pensaci due volte prima di utilizzare l'opzione `-R`, poiché è facile cambiare permessi su file e directory che in realtà non vuoi modificare, specialmente su directory con un gran numero di file e sottodirectory.

Modalità Numerica

In *modalità numerica*, i permessi sono specificati in modo diverso: come valore numerico a tre cifre in notazione ottale, un sistema numerico in base 8.

Ogni permesso ha un valore corrispondente e i permessi sono specificati nel seguente ordine: prima viene il permesso di lettura (`r`) che ha valore `4`, poi il permesso di scrittura (`w`) che ha valore `2` e infine il permesso di esecuzione (`x`), rappresentato dal valore `1`. Se non ci sono permessi, usa il valore zero (`0`). Quindi, un permesso pari a `rwx` ha valore `7` (`4+2+1`), mentre un permesso `r-x` ha

valore `5` (`4+0+1`).

La prima delle tre cifre all'interno del set dei permessi rappresenta i permessi per l'utente (`u`), la seconda rappresenta i permessi per il gruppo (`g`) e la terza rappresenta i permessi per gli altri (`o`). Se vogliamo impostare i permessi di un file a `rw-rw----`, il valore ottale da usare è `660`:

```
$ chmod 660 text.txt
$ ls -l text.txt
-rw-rw---- 1 carol carol 765 Dec 20 21:25 text.txt
```

Inoltre, la sintassi in *modalità numerica* è la stessa della *modalità simbolica*: il primo parametro rappresenta i permessi che si desidera impostare e il secondo punta al file o alla directory su cui verrà effettuata la modifica.

TIP Se il permesso ha valore *dispari*, il file è sicuramente eseguibile!

Quale sintassi usare? La *modalità numerica* è consigliata se si desidera modificare i permessi a un valore specifico, per esempio `640` (`rw- r-- ---`).

La *modalità simbolica* è più utile se si desidera modificare solo un valore specifico, indipendentemente dai permessi correnti del file. Per esempio, è possibile aggiungere solo i permessi di esecuzione per l'utente usando `chmod u+x script.sh` senza preoccuparsi, o addirittura senza neanche toccare, i permessi correnti per il gruppo e per gli altri.

Modificare la Proprietà dei File

Il comando `chown` viene utilizzato per modificare la proprietà (ownership) di un file o di una directory. La sintassi è abbastanza semplice:

```
chown username:groupname filename
```

Per esempio, controlliamo un file chiamato `text.txt`:

```
$ ls -l text.txt
-rw-rw---- 1 carol carol 1881 Dec 10 15:57 text.txt
```

L'utente che possiede il file è `carol` e anche il gruppo è `carol`. Modifichiamo ora il gruppo che possiede il file in un altro gruppo, come per esempio `students`:

```
$ chown carol:students text.txt
$ ls -l text.txt
-rw-rw---- 1 carol students 1881 Dec 10 15:57 text.txt
```

Tieni presente che l'utente che possiede il file non deve appartenere al gruppo che possiede il file stesso. Nell'esempio sopra riportato, l'utente `carol` non ha bisogno di essere un membro del gruppo `students`. Tuttavia, deve essere membro del gruppo per potergli trasferire la proprietà di gruppo del file.

L'utente o il gruppo possono essere omessi se non si desidera modificarli. Quindi, per cambiare solo il gruppo proprietario di un file, puoi usare `chown :students text.txt`. Per cambiare solo l'utente, il comando è `chown carol text.txt`. In alternativa puoi usare il comando `chgrp students text.txt` per cambiare solo il gruppo.

A meno che tu non sia amministratore di sistema (*root*), non puoi cambiare la proprietà di un file posseduto da un altro utente o da un gruppo a cui non appartieni. Se provi a farlo, riceverai il messaggio di errore `Operation not permitted` (`Operazione non consentita`).

Ricercare i Gruppi

Prima di modificare la proprietà (ownership) di un file, potrebbe essere utile sapere quali gruppi esistono nel sistema, quali utenti sono membri di un gruppo e a quali gruppi appartiene un utente. Queste attività possono essere fatte con due comandi: `groups` e `groupmems`.

Per vedere quali gruppi esistono sul tuo sistema, digita semplicemente `groups`:

```
$ groups
carol students cdrom sudo dip plugdev lpadmin sambashare
```

E se vuoi sapere a quali gruppi appartiene un utente, aggiungi il nome utente come parametro:

```
$ groups carol
carol : carol students cdrom sudo dip plugdev lpadmin sambashare
```

Per fare il contrario, mostrando quali utenti appartengono a un gruppo, usa `groupmems`. L'opzione `-g` specifica il gruppo e l'opzione `-l` elenca tutti i suoi membri:

```
$ sudo groupmems -g cdrom -l
```

```
carol
```

TIP `groupmems` può essere eseguito solo come root, l'amministratore di sistema. Se non sei attualmente loggato come root, aggiungi `sudo` prima del comando.

Permessi Speciali

Oltre ai permessi di lettura, scrittura ed esecuzione per l'utente, il gruppo e gli altri, ogni file può avere altri tre *permessi speciali* che possono alterare il modo in cui funziona una directory o come viene eseguito un programma. Possono essere specificati o in modalità simbolica o numerica e sono i seguenti:

Sticky Bit

Lo *sticky bit*, chiamato anche *flag di cancellazione limitata* (*restricted deletion flag*), ha valore ottale pari a `1` e, in modalità simbolica, è rappresentato da una `t` all'interno del set dei permessi *per gli altri*. Si applica solo alle directory e su Linux impedisce agli utenti di rimuovere o rinominare un file in una directory a meno che non possiedano quel file o quella directory.

Le directory con impostato lo sticky bit mostrano una `t` che sostituisce la `x` nel set dei permessi per gli *altri* nell'output di `ls -l`:

```
$ ls -ld Sample_Directory/
drwxr-xr-t 2 carol carol 4096 Dec 20 18:46 Sample_Directory/
```

In modalità numerica, i permessi speciali vengono specificati utilizzando una "notazione a 4 cifre" con la prima cifra che rappresenta il permesso speciale su cui agire. Per esempio, per impostare lo sticky bit (valore `1`) sulla directory `Another_Directory` in modalità numerica, con permessi `755`, puoi usare il seguente comando:

```
$ chmod 1755 Another_Directory
$ ls -ld Another_Directory
drwxr-xr-t 2 carol carol 4,0K Dec 20 18:46 Another_Directory
```

Set GID

Set GID, noto anche come SGID o bit Set Group ID, ha valore ottale pari a `2` e, in modalità simbolica, è rappresentato da una `s` nel set di permessi del *gruppo*. Può essere impostato su file eseguibili o su directory. Sui file eseguibili concede al processo risultante dall'esecuzione del file

l'accesso ai privilegi del gruppo proprietario del file. Se impostato su una directory fa in modo che ogni file o directory creata all'interno di essa erediti il gruppo dalla directory padre.

I file e le directory con impostato il bit SGID mostrano una `s` che sostituisce la `x` nel set dei permessi per il *gruppo* nell'output di `ls -l`:

```
$ ls -l test.sh
-rwxr-sr-x 1 carol carol 33 Dec 11 10:36 test.sh
```

Per aggiungere i permessi SGID a un file in modalità simbolica, puoi usare il seguente comando:

```
$ chmod g+s test.sh
$ ls -l test.sh
-rwxr-sr-x 1 carol root     33 Dec 11 10:36 test.sh
```

Il seguente esempio ti aiuterà a comprendere meglio gli effetti del bit SGID su una directory. Supponiamo di avere una directory chiamata `Sample_Directory`, di proprietà dell'utente `carol` e del gruppo `users`, con la seguente struttura di permessi:

```
$ ls -ldh Sample_Directory/
drwxr-xr-x 2 carol users 4,0K Jan 18 17:06 Sample_Directory/
```

Entriamo ora in questa directory e, usando il comando `touch`, creiamo un file vuoto al suo interno. Il risultato è il seguente:

```
$ cd Sample_Directory/
$ touch newfile
$ ls -lh newfile
-rw-r--r-- 1 carol carol 0 Jan 18 17:11 newfile
```

Come possiamo vedere il file è di proprietà dell'utente `carol` e del gruppo `carol`, ma se la directory avesse il set di permessi SGID, il risultato sarebbe diverso. Per prima cosa, aggiungiamo il bit SGID a `Sample_Directory` e controlliamo i risultati:

```
$ sudo chmod g+s Sample_Directory/
$ ls -ldh Sample_Directory/
drwxr-sr-x 2 carol users 4,0K Jan 18 17:17 Sample_Directory/
```

La `s` nei permessi del gruppo indica che è impostato il bit SGID. Entriamo ora in questa directory e, di nuovo, creiamo un file vuoto con il comando `touch`:

```
$ cd Sample_Directory/
$ touch emptyfile
$ ls -lh emptyfile
 -rw-r--r-- 1 carol users 0 Jan 18 17:20 emptyfile
```

Come possiamo vedere il gruppo che possiede il file è `users`. Questo perché il bit SGID ha fatto in modo che il file ereditasse il gruppo proprietario della sua directory padre, che è `users`.

Set UID

SUID, noto anche come Set User ID, ha valore ottale pari a `4` ed è rappresentato da una `s` nel set di permessi dell'*utente* in modalità simbolica. Si può impostare solo sui file e il suo comportamento è simile al bit SGID, ma il processo viene eseguito con i privilegi dell'*utente* che possiede il file. I file con il bit SUID mostrano una `s` che sostituisce la `x` nel set dei permessi per l'utente nell'output di `ls -l`:

```
$ ls -ld test.sh
-rwsr-xr-x 1 carol carol 33 Dec 11 10:36 test.sh
```

È possibile combinare più permessi speciali in un unico parametro sommandoli. Quindi, per impostare SGID (valore `2`) e SUID (valore `4`) in modalità numerica per lo script `test.sh` con permessi `755`, puoi usare:

```
$ chmod 6755 test.sh
```

E il risultato è:

```
$ ls -lh test.sh
-rwsr-sr-x 1 carol carol 66 Jan 18 17:29 test.sh
```

TIP

Se il tuo terminale supporta i colori, e oggigiorno la maggior parte dei terminali lo fa, puoi rapidamente vedere se sono impostati questi permessi speciali, dando uno sguardo all'output di `ls -l`. Per lo sticky bit, il nome della directory potrebbe essere visualizzato in nero con uno sfondo blu. Lo stesso vale per i file con i bit SGID (sfondo giallo) e SUID (sfondo rosso). I colori possono essere diversi a seconda della

distribuzione Linux e delle impostazioni del terminale che utilizzi.

Esercizi Guidati

1. Crea una directory chiamata `emptydir` usando il comando `mkdir emptydir`. Ora, con `ls` elenca i permessi della directory `emptydir`.

2. Crea un file vuoto chiamato `emptyfile` con il comando `touch emptyfile`. Ora, usando `chmod` con la notazione simbolica, aggiungi i permessi di esecuzione per il proprietario del file `emptyfile`, e rimuovi i permessi di scrittura ed esecuzione per tutti gli altri. Fallo usando un solo comando `chmod`.

3. Quali sono i permessi di un file chiamato `text.txt` dopo aver usato il comando `chmod 754 text.txt`?

4. Supponiamo che un file chiamato `test.sh` sia uno script di shell con i seguenti permessi e ownership:

```
-rwxr-sr-x 1 carol root     33 Dec 11 10:36 test.sh
```

   - Quali sono i permessi per il proprietario del file?
   - Se l'utente `john` esegue questo script, con quali privilegi utente verrà eseguito?
   - Usando la notazione numerica, qual è la sintassi di `chmod` per "annullare" il permesso speciale concesso a questo file?

5. Considera questo file:

```
$ ls -l /dev/sdb1
brw-rw---- 1 root disk 8, 17 Dec 21 18:51 /dev/sdb1
```

   Che tipo di file è `sdb1`? E chi può scriverci?

6. Considera i 4 file di seguito riportati:

```
drwxr-xr-t 2 carol carol 4,0K Dec 20 18:46 Another_Directory
----r--r-- 1 carol carol    0 Dec 11 10:55 foo.bar
-rw-rw-r-- 1 carol carol 1,2G Dec 20 18:22 HugeFile.zip
drwxr-sr-x 2 carol users 4,0K Jan 18 17:26 Sample_Directory
```

Indica i corrispondenti permessi per ogni file e directory utilizzando la notazione numerica a 4 cifre.

`Another_Directory`

`foo.bar`

`HugeFile.zip`

`Sample_Directory`

Esercizi Esplorativi

1. Crea un file vuoto chiamato `emptyfile` con il comando `touch emptyfile`. Ora "azzera" i permessi del file con `chmod 000 emptyfile`. Cosa succede se modifichi i permessi per `emptyfile` passando solo *un* valore per `chmod`, in notazione numerica, come per esempio `chmod 4 emptyfile`? E se ne usassimo due, come `chmod 44 emptyfile`? Cosa possiamo imparare sul modo in cui `chmod` legge il valore numerico?

2. Puoi eseguire un file per il quale hai i permessi di esecuzione, ma non quelli di lettura (`--x`)? Per quale motivo?

3. Considera i permessi per la directory temporanea `/tmp` su un sistema Linux:

```
$ ls -ld /tmp
drwxrwxrwt  19 root root  16K Dec 21 18:58 tmp
```

   L'utente, il gruppo e gli altri hanno permessi completi. Ma può un utente ordinario eliminare *qualsiasi file* all'interno di questa directory? Per quale motivo?

4. Un file chiamato `test.sh` ha i seguenti permessi: `-rwsr-xr-x`, il che significa che è impostato il bit SUID. Ora esegui i seguenti comandi:

```
$ chmod u-x test.sh
$ ls -l test.sh
-rwSr-xr-x 1 carol carol 33 Dec 11 10:36 test.sh
```

   Che cosa abbiamo fatto? Cosa significa la `S` maiuscola?

5. Come puoi creare una directory chiamata `Box` in cui tutti i file sono automaticamente di proprietà del gruppo `users` e possono essere eliminati solo dall'utente che li ha creati?

Sommario

Essendo un sistema multiutente, Linux ha bisogno di un modo per tenere traccia di chi possiede e di chi può accedere a ogni file. Questo viene realizzato attraverso un sistema di permessi a tre livelli e in questa lezione abbiamo imparato tutto su come funziona questo sistema.

In questa lezione hai imparato come usare `ls` per ottenere informazioni sui permessi dei file, come controllare o cambiare chi può creare, eliminare o modificare un file con `chmod`, sia usando la notazione *numerica* sia usando la notazione *simbolica* e come cambiare la proprietà (ownership) dei file con `chown` e `chgrp`.

In questa lezione sono stati discussi i seguenti comandi:

`ls`
: Elenca i file, eventualmente includendo dettagli come, per esempio, i permessi.

`chmod`
: Modifica i permessi di un file o di una directory.

`chown`
: Modifica l'utente e/o il gruppo proprietario di un file o di una directory.

`chgrp`
: Modifica il gruppo proprietario di un file o di una directory.

Risposte agli Esercizi Guidati

1. Crea una directory chiamata `emptydir` usando il comando `mkdir emptydir`. Ora, con `ls` elenca i permessi della directory `emptydir`.

   Aggiungi l'opzione `-d` a `ls` per vedere gli attributi di file di una directory, invece di elencarne il contenuto. Quindi la risposta è:

   ```
   ls -l -d emptydir
   ```

   Ottieni punti bonus se hai unito le due opzioni in una sola, come in `ls -ld emptydir`.

2. Crea un file vuoto chiamato `emptyfile` con il comando `touch emptyfile`. Ora, usando `chmod` con la notazione simbolica, aggiungi i permessi di esecuzione per il proprietario del file `emptyfile`, e rimuovi i permessi di scrittura ed esecuzione per tutti gli altri. Fallo usando un solo comando `chmod`.

   Pensa in questo modo:

   - "Per l'utente che possiede il file (`u`) aggiungi (`+`) i permessi di esecuzione (`x`)", quindi `u+x`.
   - "Per il gruppo (`g`) e per gli altri utenti (`o`), rimuovi (`-`) i permessi di scrittura (`w`) ed esecuzione (`x`)", quindi `go-wx`.

     Per combinare questi due set di permessi, aggiungiamo una virgola tra di loro. Quindi il risultato finale è:

     ```
     chmod u+x,go-wx emptyfile
     ```

3. Quali sono i permessi di un file chiamato `text.txt` dopo aver usato il comando `chmod 754 text.txt`?

   Ricorda che nella notazione numerica ogni cifra rappresenta un insieme di tre permessi, ognuno con un rispettivo valore: per la *lettura* il valore è `4`, per la *scrittura* il valore è `2`, per l'*esecuzione* il valore è `1`, mentre nessun permesso ha valore `0`. Possiamo ottenere il valore di una cifra sommando i corrispondenti valori per ogni permesso: `7` è `4+2+1` o `rwx`, `5` è `4+0+1` o `r-x` e `4` indica permessi di sola lettura o `r--`. Quindi i permessi per `text.txt` sono:

   ```
   rwxr-xr--
   ```

4. Supponiamo che un file chiamato `test.sh` sia uno script di shell con i seguenti permessi e ownership:

```
-rwxr-sr-x 1 carol root     33 Dec 11 10:36 test.sh
```

   - Quali sono i permessi per il proprietario del file?

     I permessi per il proprietario (dal secondo al quarto carattere nell'output di `ls -l`) sono `rwx`; quindi la risposta è: "leggere, scrivere ed eseguire il file".

   - Se l'utente `john` esegue questo script, con quali privilegi utente verrà eseguito?

     Presta attenzione ai permessi per il *gruppo*. Questi sono `r-s`, il che significa che è impostato il bit SGID. Il gruppo che possiede questo file è `root`; quindi lo script, anche quando viene avviato da un utente ordinario, verrà eseguito con i privilegi di root.

   - Usando la notazione numerica, qual è la sintassi di `chmod` per "annullare" il permesso speciale concesso a questo file?

     Possiamo "annullare" i permessi speciali passando una quarta cifra, `0`, a `chmod`. I permessi correnti sono `755`; quindi il comando è `chmod 0755`.

5. Considera questo file:

```
$ ls -l /dev/sdb1
brw-rw---- 1 root disk 8, 17 Dec 21 18:51 /dev/sdb1
```

   Che tipo di file è `sdb1`? E chi può scriverci?

   Il primo carattere dell'output di `ls -l` mostra il tipo di file. `b` indica un *dispositivo a blocchi*, solitamente un disco (interno o esterno), connesso alla macchina. Il proprietario (`root`) e qualsiasi utente del gruppo `disk` possono scrivere su di esso.

6. Considera i 4 file di seguito riportati:

```
drwxr-xr-t 2 carol carol 4,0K Dec 20 18:46 Another_Directory
----r--r-- 1 carol carol    0 Dec 11 10:55 foo.bar
-rw-rw-r-- 1 carol carol 1,2G Dec 20 18:22 HugeFile.zip
drwxr-sr-x 2 carol users 4,0K Jan 18 17:26 Sample_Directory
```

   Indica i corrispondenti permessi per ogni file e directory utilizzando la notazione numerica a 4

cifre.

I corrispondenti permessi, in notazione numerica, sono i seguenti:

`Another_Directory`

Soluzione: `1755`

`1` per lo sticky bit, `755` per i normali permessi (`rwx` per l'utente, `r-x` per il gruppo e per gli altri).

`foo.bar`

Soluzione: `0044`

Nessun permesso speciale (quindi la prima cifra è `0`), nessun permesso per l'utente (`---`) e permessi di sola lettura (`r—r--`) per il gruppo e per gli altri.

`HugeFile.zip`

Soluzione: `0664`

Nessun permesso speciale, quindi la prima cifra è `0`. `6` (`rw-`) per l'utente e il gruppo, `4` (`r--`) per gli altri.

`Sample_Directory`

Soluzione: `2755`

`2` per il bit SGID, `7` (`rwx`) per l'utente, `5` (`r-x`) per il gruppo e per gli altri.

Risposte agli Esercizi Esplorativi

1. Crea un file vuoto chiamato `emptyfile` con il comando `touch emptyfile`. Ora "azzera" i permessi del file con `chmod 000 emptyfile`. Cosa succede se modifichi i permessi per `emptyfile` passando solo *un* valore per `chmod`, in notazione numerica, come per esempio `chmod 4 emptyfile`? E se ne usassimo due, come `chmod 44 emptyfile`? Cosa possiamo imparare sul modo in cui `chmod` legge il valore numerico?

   Ricorda che abbiamo "azzerato" i permessi di `emptyfile`. Quindi, il suo stato iniziale è:

   ```
   ---------- 1 carol carol    0 Dec 11 10:55 emptyfile
   ```

   Ora proviamo il primo comando: `chmod 4 emptyfile`:

   ```
   $ chmod 4 emptyfile
   $ ls -l emptyfile
   -------r-- 1 carol carol 0 Dec 11 10:55 emptyfile
   ```

   I permessi per *gli altri* sono stati modificati. E se provassimo due cifre, come `chmod 44 emptyfile`?

   ```
   $ chmod 44 emptyfile
   $ ls -l emptyfile
   ----r--r-- 1 carol carol 0 Dec 11 10:55 emptyfile
   ```

   Ora i permessi per il *gruppo* e per gli *altri* sono stati alterati. Da questo possiamo dedurre che in notazione numerica `chmod` legge il valore "al contrario", dalla cifra meno significativa (*gli altri*) a quella più significativa (*l'utente*). Se passi una cifra, modifichi i permessi per gli *altri*. Con due cifre modifichi i permessi per il *gruppo* e per gli *altri*, con tre modifichi i permessi per l'*utente*, il *gruppo* e gli *altri*, e con quattro cifre modifichi i permessi per l'*utente*, il *gruppo*, gli *altri* e i permessi speciali.

2. Puoi eseguire un file per il quale hai i permessi di esecuzione, ma non quelli di lettura (`--x`)? Per quale motivo?

   Di primo acchito la risposta sembra ovvia: se hai il permesso di esecuzione, il file dovrebbe essere eseguito. Questo vale per i programmi in formato binario che vengono eseguiti direttamente dal kernel. Tuttavia, ci sono programmi (ad esempio gli script di shell) che devono essere prima letti e interpretati; quindi in questi casi è necessario impostare anche il permesso

di lettura (`r`).

3. Considera i permessi per la directory temporanea `/tmp` su un sistema Linux:

```
$ ls -ld /tmp
drwxrwxrwt  19 root root  16K Dec 21 18:58 tmp
```

L'utente, il gruppo e gli altri hanno permessi completi. Ma può un utente ordinario eliminare *qualsiasi file* all'interno di questa directory? Per quale motivo?

`/tmp` è ciò che chiamiamo una *directory scrivibile da chiunque (world writeable)*, il che significa che qualsiasi utente può scriverci. Ma non vogliamo che un utente modifichi i file creati da altri, quindi è impostato lo *sticky bit* (come indicato dalla `t` nei permessi per gli *altri*). Ciò significa che un utente può eliminare i file in `/tmp`, ma solo se ha creato quei file.

4. Un file chiamato `test.sh` ha i seguenti permessi: `-rwsr-xr-x`, il che significa che è impostato il bit SUID. Ora esegui i seguenti comandi:

```
$ chmod u-x test.sh
$ ls -l test.sh
-rwSr-xr-x 1 carol carol 33 Dec 11 10:36 test.sh
```

Che cosa abbiamo fatto? Cosa significa la `S` maiuscola?

Abbiamo rimosso i permessi di esecuzione per l'utente proprietario del file. La `s` (o la `t`) prende il posto della `x` nell'output di `ls -l`; quindi il sistema ha bisogno di un modo per mostrare se l'utente ha i permessi di esecuzione o meno. Lo fa mettendo il carattere speciale in maiuscolo o minuscolo.

Una `s` minuscola nel primo gruppo di permessi significa che l'utente che possiede il file ha i permessi di esecuzione e che è impostato il bit SUID. Una `S` maiuscola significa che l'utente che possiede il file non ha (`-`) i permessi di esecuzione e che è impostato il bit SUID.

Lo stesso si può dire per SGID. Una `s` minuscola nel secondo gruppo di permessi significa che il gruppo che possiede il file ha i permessi di esecuzione e che è impostato il bit SGID. Una `S` maiuscola significa che il gruppo che possiede il file non ha (`-`) i permessi di esecuzione e che è impostato il bit SGID.

Questo vale anche per lo sticky bit, rappresentato da una `t` nel terzo gruppo di permessi. Una `t` minuscola significa che è impostato lo sticky bit e che gli altri hanno i permessi di esecuzione. Una `T` maiuscola significa che è impostato lo sticky bit e che gli altri non hanno i permessi di

esecuzione.

5. Come puoi creare una directory chiamata `Box` in cui tutti i file sono automaticamente di proprietà del gruppo `users` e possono essere eliminati solo dall'utente che li ha creati?

   Questo è un processo a più fasi. Il primo passo consiste nel creare la directory:

   ```
   $ mkdir Box
   ```

   Vogliamo che ogni file creato all'interno di questa directory venga automaticamente assegnato al gruppo `users`. Possiamo farlo impostando questo gruppo come proprietario della directory e poi impostando il bit SGID sulla directory stessa. Dobbiamo anche assicurarci che qualsiasi membro del gruppo possa scrivere in questa directory.

   Dato che non abbiamo bisogno di preoccuparci degli altri permessi e vogliamo solo "attivare" i bit speciali, ha senso usare la modalità simbolica:

   ```
   $ chown :users Box/
   $ chmod g+wxs Box/
   ```

   Nota che se l'utente corrente non appartiene al gruppo `users`, dovrai usare il comando `sudo` prima dei comandi sopra riportati per apportare la modifica come root.

   Ora per la parte finale, assicuriamoci che solo l'utente che ha creato un file sia autorizzato a eliminarlo. Questo viene fatto impostando lo sticky bit (rappresentato da una `t`) sulla directory. Ricorda che è impostato sul set dei permessi per gli altri (`o`).

   ```
   $ chmod o+t Box/
   ```

   I permessi sulla directory `Box` dovrebbero apparire nel seguente modo:

   ```
   drwxrwsr-t 2 carol users  4,0K Jan 18 19:09 Box
   ```

   Ovviamente, puoi specificare SGID e sticky bit usando un solo comando `chmod`:

   ```
   $ chmod g+wxs,o+t Box/
   ```

   Ottieni dei punti bonus se ci hai pensato.

5.4 Directory e File Speciali

Obiettivi LPI di riferimento

Linux Essentials v1.6, Exam 010, Objective 5.4

Peso

1

Aree di Conoscenza Chiave

- Utilizzo di file e directory temporanei
- Link simbolici

Di seguito è riportato un elenco parziale dei file, dei termini e dei comandi utilizzati

- `/tmp/`, `/var/tmp/` e Sticky Bit
- `ls -d`
- `ln -s`

5.4 Lezione 1

Certificazione:	Linux Essentials
Versione:	1.6
Argomento:	5 Sicurezza e Permessi sui File
Obiettivo:	5.4 Directory e File Speciali
Lezione:	1 di 1

Introduzione

In Linux, tutto viene trattato come un file. Tuttavia alcuni file ricevono un trattamento speciale, sia per dove sono memorizzati, come nel caso dei file temporanei, sia per il modo in cui interagiscono con il *filesystem*, come nel caso dei link. In questa lezione impareremo posizione, funzionamento e gestione di tali file.

File Temporanei

I file temporanei sono file utilizzati dai programmi per archiviare dati necessari solo per un periodo di tempo limitato: per esempio dati dei processi in esecuzione, log di guasti (*crash logs*), file *scratch* di un salvataggio automatico, file intermedi utilizzati durante una conversione di file, file di *cache* e così via.

Posizione dei File Temporanei

La versione 3.0 del *Filesystem Hierarchy Standard* (FHS) definisce le posizioni standard per i file temporanei sui sistemi Linux. Ogni posizione ha uno scopo e un comportamento diverso: è consigliato agli sviluppatori seguire le convenzioni impostate dal FHS quando scrivono dati

temporanei sul disco.

`/tmp`

Secondo il FHS, i programmi non dovrebbero supporre che i file scritti qui vengano conservati tra invocazioni di un programma. Il *consiglio* è di pulire questa directory (tutti i file cancellati) durante l'avvio del sistema, sebbene questo *non sia obbligatorio*.

`/var/tmp`

Un'altra posizione per i file temporanei, ma questa *non dovrebbe essere pulita* durante l'avvio del sistema, ovvero i file archiviati qui di solito persistono tra i riavvii.

`/run`

Questa directory contiene dati variabili di run-time utilizzati dai processi in esecuzione, come i file di identificativo di processo (`.pid`). I programmi che richiedono più di un file di run-time possono creare sottodirectory qui. Questa posizione *deve essere pulita* durante l'avvio del sistema. In passato la funzione di questa directory era assolta da `/var/run`, e su alcuni sistemi `/var/run` potrebbe essere un link simbolico a `/run`.

Nota che nulla impedisce a un programma di creare file temporanei altrove nel sistema, ma è buona norma rispettare le convenzioni stabilite dal FHS.

Permessi sui File Temporanei

L'esistenza di directory temporanee a livello di sistema su un sistema multiutente presenta alcune criticità in relazione ai permessi di accesso. Di primo acchito si potrebbe pensare che tali directory siano "scrivibili da chiunque" ("world-writable"), cioè che qualsiasi utente possa scrivere o cancellare dati al loro interno. Ma se fosse vero, come potremmo impedire a un utente di cancellare o modificare file creati da un altro utente?

La soluzione è un permesso speciale chiamato *sticky bit*, che si applica sia alle directory sia ai file. Tuttavia, per motivi di sicurezza, il kernel Linux ignora lo sticky bit quando viene applicato ai file. Quando questo bit speciale è impostato su una directory, impedisce agli utenti di rimuovere o rinominare un file all'interno di quella directory a meno che non siano i proprietari del file stesso.

Le directory con impostato lo sticky bit mostrano una `t` che sostituisce la `x` nei permessi per gli *altri* nell'output di `ls -l`. Per esempio, controlliamo i permessi per le directory `/tmp` e `/var/tmp`:

```
$ ls -ldh /tmp/ /var/tmp/
drwxrwxrwt 25 root root 4,0K Jun  7 18:52 /tmp/
drwxrwxrwt 16 root root 4,0K Jun  7 09:15 /var/tmp/
```

Come puoi vedere dalla `t` che sostituisce la `x` nei permessi per gli *altri*, entrambe le directory hanno impostato lo sticky bit.

Per impostare lo sticky bit su una directory utilizzando `chmod` in modalità numerica, utilizza la notazione a quattro cifre e `1` come prima cifra. Per esempio:

```
$ chmod 1755 temp
```

Questo comando imposta lo sticky bit sulla directory `temp` e i imposta permessi a `rwxr-xr-t`.

Quando si utilizza la modalità simbolica, è necessario utilizzare il parametro `t`. Quindi utilizza `+t` per impostare lo sticky bit e `-t` per rimuoverlo, nel seguente modo:

```
$ chmod +t temp
```

Comprendere i Link

Abbiamo già detto che su Linux tutto viene trattato come un file, ma ne esiste un tipo *speciale*, chiamato *link*. Ci sono due tipi di link su un sistema Linux:

Link Simbolici

Chiamati anche *soft link*, puntano al percorso di un altro file. Se elimini il file a cui punta il link (chiamato *target*) il link esisterà ancora, ma "smetterà di funzionare", poiché adesso punta al "nulla".

Hard link

Pensa un hard link come un secondo nome del file originale. *Non* sono duplicati, ma voci aggiuntive nel file system che puntano alla stessa posizione (*inode*) sul disco.

TIP

Un *inode* è una struttura dati che memorizza gli attributi di un oggetto (come un file o una directory) su un filesystem. Tra questi attributi ci sono il nome del file, i permessi, l'ownership e su quali blocchi del disco sono memorizzati i dati dell'oggetto. Pensalo come una voce in un indice: da qui il nome, che deriva da "index node".

Lavorare con gli Hard Link

Creare Hard Link

Il comando per creare un hard link su Linux è `ln`. La sintassi base è:

```
$ ln TARGET LINK_NAME
```

Il `TARGET` deve già esistere (questo è il file a cui punta il link), e, se il target non si trova nella directory corrente, o se si desidera creare il link altrove, *è necessario* specificarne il percorso completo. Per esempio, il comando

```
$ ln target.txt /home/carol/Documents/hardlink
```

crea un file chiamato `hardlink` nella directory `/home/carol/Documents/`, collegato al file `target.txt` nella directory corrente.

Se tralasci l'ultimo parametro (`LINK_NAME`), viene creato un link con lo stesso nome del target nella directory corrente.

Gestire gli Hard Link

Gli hard link sono voci nel filesystem che hanno nomi diversi, ma che puntano agli stessi dati sul disco. Tutti questi nomi sono equivalenti e possono essere utilizzati per fare riferimento a un file. Se si modifica il contenuto di uno dei nomi, viene modificato il contenuto di tutti gli altri nomi che puntano a quel file, poiché tutti questi nomi puntano agli stessi dati. Se elimini uno dei nomi, gli altri continueranno a funzionare.

Ciò accade perché quando si "elimina" un file i dati non vengono effettivamente cancellati dal disco. Il sistema cancella semplicemente la voce nella tabella del filesystem che punta all'inode corrispondente ai dati sul disco. Ma se c'è una seconda voce che punta allo stesso inode, sarà comunque possibile accedere ai dati. Pensa a questo come a due strade che convergono nello stesso punto. Anche se blocchi o devii una delle strade, puoi comunque raggiungere la destinazione utilizzando l'altra.

Puoi verificarlo utilizzando l'opzione `-i` di `ls`. Considera il seguente contenuto di una directory:

```
$ ls -li
total 224
3806696 -r--r--r-- 2 carol carol 111702 Jun  7 10:13 hardlink
3806696 -r--r--r-- 2 carol carol 111702 Jun  7 10:13 target.txt
```

Il numero prima dei permessi è il numero di inode. Hai notato che sia il file `hardlink` sia il file `target.txt` hanno lo stesso numero (`3806696`)? Questo perché uno è un hard link dell'altro.

Ma qual è l'originale e qual è il link? Non si può davvero dire, poiché, a tutti gli effetti, sono la

stessa cosa.

Nota che ogni hard link che punta a un file incrementa il *contatore di link* (*link count*) del file. Questo è il numero subito dopo i permessi nell'output di `ls -l`. Per impostazione predefinita, ogni file ha un link count pari a `1` (`2` per le directory) e ogni hard link a esso aumenta il conteggio di uno. Quindi, questo è il motivo per cui nei file dell'elenco sopra riportato il link count è pari a `2`.

A differenza dei link simbolici, è possibile creare solo hard link ai file; sia il link sia il target devono risiedere nello stesso file system.

Spostare e Rimuovere Hard Link

Poiché gli hard link sono trattati come file normali, possono essere cancellati con `rm` e rinominati o spostati nel filesystem con `mv`. E poiché un hard link punta allo stesso inode del target, può essere spostato liberamente, senza timore di "rompere" il collegamento.

Link Simbolici

Creare Link Simbolici

Anche il comando usato per creare un link simbolico è `ln`, ma con l'aggiunta dell'opzione `-s`, nel seguente modo:

```
$ ln -s target.txt /home/carol/Documents/softlink
```

Questo comando crea un file chiamato `softlink` nella directory `/home/carol/Documents/` che punta al file `target.txt` nella directory corrente.

Come per gli hard link è possibile omettere il nome del link per creare un link con lo stesso nome del target nella directory corrente.

Gestire i Link Simbolici

I link simbolici puntano a un altro percorso nel filesystem. È possibile creare soft link di file *e* directory, anche su partizioni diverse. È abbastanza facile individuare un link simbolico dall'output di `ls`:

```
$ ls -lh
total 112K
-rw-r--r-- 1 carol carol 110K Jun  7 10:13 target.txt
```

```
lrwxrwxrwx 1 carol carol   12 Jun  7 10:14 softlink -> target.txt
```

Nell'esempio sopra riportato, il primo carattere nei permessi del file `softlink` è `l`, che indica un link simbolico. Inoltre, subito dopo il nome del file possiamo notare il nome del target a cui punta il link: il file `target.txt`.

Nota che negli elenchi di file e directory, i soft link mostrano sempre i permessi `rwx` per l'utente, il gruppo e gli altri, ma in pratica i loro permessi di accesso sono gli stessi di quelli del target.

Spostare e Rimuovere Link Simbolici

Come per gli hard link, i link simbolici possono essere rimossi usando `rm` e possono essere spostati o rinominati usando `mv`. Tuttavia, è necessario prestare particolare attenzione durante la loro creazione per evitare di "rompere" il link nel caso in cui venga spostato dalla sua posizione originale.

Quando si creano link simbolici è necessario essere consapevoli del fatto che, a meno che un percorso non sia completamente specificato, la posizione del target viene interpretata come *relativa* alla posizione del link. Ciò potrebbe creare problemi se il link o il file a cui punta viene spostato.

È più facile capirlo con un esempio: supponiamo di avere un file chiamato `original.txt` nella directory corrente e di voler creare un link simbolico a esso chiamato `softlink`. Possiamo usare:

```
$ ln -s original.txt softlink
```

E a quanto pare tutto è andato per il verso giusto. Controlliamo con `ls`:

```
$ ls -lh
total 112K
-r--r--r-- 1 carol carol 110K Jun  7 10:13 original.txt
lrwxrwxrwx 1 carol carol   12 Jun  7 19:23 softlink -> original.txt
```

Guarda come è costruito il link: `softlink` punta a (→) `original.txt`. Tuttavia, vediamo cosa succede se spostiamo il link nella directory padre e proviamo a visualizzarne il contenuto utilizzando il comando `less`:

```
$ mv softlink ../
$ less ../softlink
```

```
../softlink: No such file or directory
```

Poiché il percorso di `original.txt` non è stato specificato, il sistema presume che si trovi nella stessa directory del link. Quando questo non è più vero, il link smette di funzionare.

Per evitare ciò è necessario specificare sempre il percorso completo del target durante la creazione del link:

```
$ ln -s /home/carol/Documents/original.txt softlink
```

In questo modo, indipendentemente da dove sposti il link, questo continuerà a funzionare, perché punta alla posizione assoluta del target. Controlla con `ls`:

```
$ ls -lh
total 112K
lrwxrwxrwx 1 carol carol   40 Jun  7 19:34 softlink -> /home/carol/Documents/original.txt
```

Esercizi Guidati

1. Immagina che un programma debba creare un file temporaneo monouso che non sarà mai più necessario dopo la chiusura del programma. Qual è la directory corretta in cui creare questo file?

2. Qual è la directory temporanea che *deve* essere cancellata durante il processo di avvio?

3. Qual è il parametro di `chmod` per abilitare, in modalità *simbolica*, lo sticky bit su una directory?

4. Immagina che ci sia un file chiamato `document.txt` nella directory `/home/carol/Documents`. Qual è il comando per creare un link simbolico, chiamato `text.txt`, a tale file nella directory corrente?

5. Spiega la differenza tra un hard link e una copia di un file.

Esercizi Esplorativi

1. Immagina di creare un file chiamato `recipes.txt` all'interno di una directory . All'interno di questa directory, crei anche un hard link a questo file, chiamato `receitas.txt`, e un link simbolico (o *soft*) a questo chiamato `rezepte.txt`.

   ```
   $ touch recipes.txt
   $ ln recipes.txt receitas.txt
   $ ln -s receitas.txt rezepte.txt
   ```

   Il contenuto della directory dovrebbe apparire così:

   ```
   $ ls -lhi
   total 160K
   5388833 -rw-r--r-- 4 carol carol 77K jun 17 17:25 receitas.txt
   5388833 -rw-r--r-- 4 carol carol 77K jun 17 17:25 recipes.txt
   5388837 lrwxrwxrwx 1 carol carol  12 jun 24 10:12 rezepte.txt -> receitas.txt
   ```

   Ricorda che, in quanto hard link, `receitas.txt` punta allo stesso inode di `recipes.txt`. Che cosa succede al soft link `rezepte.txt` se `receitas.txt` venisse cancellato? Per quale motivo?

2. Immagina di avere una chiavetta USB collegata al tuo sistema montata su `/media/youruser/FlashA`. Vuoi creare nella tua directory home un link chiamato `schematics.pdf` che punti al file `esquema.pdf` nella directory principale della chiavetta. Digiti quindi il comando:

   ```
   $ ln /media/youruser/FlashA/esquema.pdf ~/schematics.pdf
   ```

   Che cosa succede? Per quale motivo?

3. Considera il seguente output di `ls -lah`:

   ```
   $ ls -lah
   total 3,1M
   drwxr-xr-x 2 carol carol 4,0K jun 17 17:27 .
   drwxr-xr-x 5 carol carol 4,0K jun 17 17:29 ..
   -rw-rw-r-- 1 carol carol 2,8M jun 17 15:45 compressed.zip
   -rw-r--r-- 4 carol carol  77K jun 17 17:25 document.txt
   -rw-rw-r-- 1 carol carol 216K jun 17 17:25 image.png
   ```

```
-rw-r--r-- 4 carol carol  77K jun 17 17:25 text.txt
```

- Quanti link puntano al file `document.txt`?
- Sono soft o hard link?
- Quale parametro devi passare a `ls` per vedere quale inode occupa ogni file?

4. Immagina di avere nella tua directory `~/Documents` un file chiamato `clients.txt` contenente alcuni nomi di client e una directory chiamata `somedir`. All'interno di essa c'è un *altro* file *anch'esso* chiamato `clients.txt` con nomi differenti. Per replicare questa struttura utilizza i seguenti comandi.

```
$ cd ~/Documents
$ echo "John, Michael, Bob" > clients.txt
$ mkdir somedir
$ echo "Bill, Luke, Karl" > somedir/clients.txt
```

Crea poi un link all'interno di `somedir` chiamato `partners.txt` che punta a questo file, utilizzando i seguenti comandi:

```
$ cd somedir/
$ ln -s clients.txt partners.txt
```

Quindi la struttura della directory è:

```
Documents
|-- clients.txt
`-- somedir
    |-- clients.txt
    `-- partners.txt -> clients.txt
```

Ora, sposta `partners.txt` da `somedir` a `~/Documents` ed elencane il contenuto.

```
$ cd ~/Documents/
$ mv somedir/partners.txt .
```

```
$ less partners.txt
```

Il link funziona ancora? In tal caso, quale file avrà il suo contenuto elencato? Per quale motivo?

5. Considera i seguenti file:

```
-rw-r--r-- 1 carol carol 19 Jun 24 11:12 clients.txt
lrwxrwxrwx 1 carol carol 11 Jun 24 11:13 partners.txt -> clients.txt
```

Quali sono i permessi di accesso per `partners.txt`? Per quale motivo?

Sommario

In questa lezione hai imparato:

- Dove vengono memorizzati i file temporanei;
- Qual è il permesso speciale che viene loro applicato;
- Cosa sono i link;
- La differenza tra link *simbolici* e *hard* link;
- Come creare i link;
- Come spostarli, rinominarli o rimuoverli.

In questa lezione sono stati discussi i seguenti comandi:

- `ln`
- Il parametro `-i` di `ls`

Risposte agli Esercizi Guidati

1. Immagina che un programma debba creare un file temporaneo monouso che non sarà mai più necessario dopo la chiusura del programma. Qual è la directory corretta in cui creare questo file?

   Poichè, dopo che il programma ha terminato l'esecuzione, non ci interessa più il file, la directory corretta è `/tmp`.

2. Qual è la directory temporanea che *deve* essere cancellata durante il processo di avvio?

   La directory è `/run` o, su alcuni sistemi, `/var/run`.

3. Qual è il parametro di `chmod` per abilitare, in modalità *simbolica*, lo sticky bit su una directory?

   Il simbolo dello sticky bit in modalità simbolica è `t`. Dato che vogliamo abilitare (aggiungere) questo permesso alla directory, il parametro è `+t`.

4. Immagina che ci sia un file chiamato `document.txt` nella directory `/home/carol/Documents`. Qual è il comando per creare un link simbolico, chiamato `text.txt`, a tale file nella directory corrente?

   `ln -s` è il comando per creare un link simbolico. Poiché è necessario specificare il percorso completo del file del quale stai creando il link, il comando è:

   ```
   $ ln -s /home/carol/Documents/document.txt text.txt
   ```

5. Spiega la differenza tra un hard link e una copia di un file.

   Un hard link è solo un altro nome per un file. Anche se sembra un duplicato del file originale, a tutti gli effetti sia il link sia il file originale sono la stessa cosa, poiché puntano agli stessi dati sul disco. Le modifiche apportate al contenuto del link si riflettono sul file originale e viceversa. Una copia è un'entità completamente indipendente, che occupa un posto diverso sul disco. Le modifiche alla copia non si riflettono sull'originale e viceversa.

Risposte agli Esercizi Esplorativi

1. Immagina di creare un file chiamato `recipes.txt` all'interno di una directory . All'interno di questa directory, crei anche un hard link a questo file, chiamato `receitas.txt`, e un link simbolico (o *soft*) a questo chiamato `rezepte.txt`.

   ```
   $ touch recipes.txt
   $ ln recipes.txt receitas.txt
   $ ln -s receitas.txt rezepte.txt
   ```

   Il contenuto della directory dovrebbe apparire così:

   ```
   $ ls -lhi
   total 160K
   5388833 -rw-r--r-- 4 carol carol 77K jun 17 17:25 receitas.txt
   5388833 -rw-r--r-- 4 carol carol 77K jun 17 17:25 recipes.txt
   5388837 lrwxrwxrwx 1 carol carol  12 jun 24 10:12 rezepte.txt -> receitas.txt
   ```

   Ricorda che, in quanto hard link, `receitas.txt` punta allo stesso inode di `recipes.txt`. Che cosa succede al soft link `rezepte.txt` se `receitas.txt` venisse cancellato? Per quale motivo?

   Il soft link `rezepte.txt` smetterebbe di funzionare. Questo perché i soft link puntano ai nomi, non agli inode, e il nome `receitas.txt` non esiste più, anche se i dati sono ancora sul disco con il nome `recipes.txt`.

2. Immagina di avere una chiavetta USB collegata al tuo sistema montata su `/media/youruser/FlashA`. Vuoi creare nella tua directory home un link chiamato `schematics.pdf` che punti al file `esquema.pdf` nella directory principale della chiavetta. Digiti quindi il comando:

   ```
   $ ln /media/youruser/FlashA/esquema.pdf ~/schematics.pdf
   ```

   Che cosa succede? Per quale motivo?

   Il comando non viene eseguito correttamente. Il messaggio di errore è `Invalid cross-device link` facendoti capire il motivo: gli hard link non possono puntare a un target in una diversa partizione o in un diverso dispositivo. L'unico modo per creare un link come questo è usare un link *simbolico* o *soft*, aggiungendo il parametro `-s` a `ln`.

3. Considera il seguente output di `ls -lah`:

```
$ ls -lah
total 3,1M
drwxr-xr-x 2 carol carol 4,0K jun 17 17:27 .
drwxr-xr-x 5 carol carol 4,0K jun 17 17:29 ..
-rw-rw-r-- 1 carol carol 2,8M jun 17 15:45 compressed.zip
-rw-r--r-- 4 carol carol  77K jun 17 17:25 document.txt
-rw-rw-r-- 1 carol carol 216K jun 17 17:25 image.png
-rw-r--r-- 4 carol carol  77K jun 17 17:25 text.txt
```

- Quanti link puntano al file `document.txt`?

  Ogni file inizia con un link count pari a `1`. Poiché il link count del file è `4`, ci sono tre link che puntano a quel file.

- Sono soft o hard link?

  Sono hard link, poiché i soft link non incrementano il link count di un file.

- Quale parametro devi passare a `ls` per vedere quale inode occupa ogni file?

  Il parametro è `-i`. L'inode viene mostrato come prima colonna nell'output di `ls`, come puoi vedere nel seguente esempio:

  ```
  $ ls -lahi
  total 3,1M
  5388773 drwxr-xr-x 2 rigues rigues 4,0K jun 17 17:27 .
  5245554 drwxr-xr-x 5 rigues rigues 4,0K jun 17 17:29 ..
  5388840 -rw-rw-r-- 1 rigues rigues 2,8M jun 17 15:45 compressed.zip
  5388833 -rw-r--r-- 4 rigues rigues  77K jun 17 17:25 document.txt
  5388837 -rw-rw-r-- 1 rigues rigues 216K jun 17 17:25 image.png
  5388833 -rw-r--r-- 4 rigues rigues  77K jun 17 17:25 text.txt
  ```

4. Immagina di avere nella tua directory `~/Documents` un file chiamato `clients.txt` contenente alcuni nomi di client e una directory chiamata `somedir`. All'interno di essa c'è un *altro* file *anch'esso* chiamato `clients.txt` con nomi differenti. Per replicare questa struttura utilizza i seguenti comandi.

   ```
   $ cd ~/Documents
   $ echo "John, Michael, Bob" > clients.txt
   $ mkdir somedir
   $ echo "Bill, Luke, Karl" > somedir/clients.txt
   ```

Crea poi un link all'interno di `somedir` chiamato `partners.txt` che punta a questo file, utilizzando i seguenti comandi:

```
$ cd somedir/
$ ln -s clients.txt partners.txt
```

Quindi la struttura della directory è:

```
Documents
|-- clients.txt
`-- somedir
    |-- clients.txt
    `-- partners.txt -> clients.txt
```

Ora, sposta `partners.txt` da `somedir` a `~/Documents` ed elencane il contenuto.

```
$ cd ~/Documents/
$ mv somedir/partners.txt .
$ less partners.txt
```

Il link funziona ancora? In tal caso, quale file avrà il suo contenuto elencato? Per quale motivo?

È "ingannevole", ma il link funziona e il file elencato sarà quello in `~/Documents`, contenente i nomi `John`, `Michael`, `Bob`.

Ricorda che poiché non hai specificato il percorso completo del target `clients.txt` durante la creazione del soft link `partners.txt`, il percorso del target viene interpretato come relativo alla posizione del link, che in questo caso è la directory corrente.

Quando il link viene spostato da `~/Documents/somedir` a `~/Documents`, dovrebbe smettere di funzionare, poiché il target non è più nella stessa directory del link. Tuttavia, per coincidenza, c'è un file chiamato `clients.txt` in `~/Documents`: quindi il link punterà a questo file, invece che al target originale all'interno di `~/somedir`.

Per evitare ciò, specifica sempre il percorso completo del target quando crei un link simbolico.

5. Considera i seguenti file:

```
-rw-r--r-- 1 rigues rigues 19 Jun 24 11:12 clients.txt
lrwxrwxrwx 1 rigues rigues 11 Jun 24 11:13 partners.txt -> clients.txt
```

Quali sono i permessi di accesso per `partners.txt`? Per quale motivo?

I permessi di accesso di `partners.txt` sono `rw-r--r--`, poiché i link ereditano sempre gli stessi permessi di accesso del target.

Imprint

© 2025 by Linux Professional Institute: Learning Materials, “Linux Essentials (Versione 1.6)”.

PDF generato: 2025-07-14

Questa opera è concessa in licenza con Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND 4.0). Per visualizzare una copia di questa licenza, visitare

https://creativecommons.org/licenses/by-nc-nd/4.0/

Sebbene Linux Professional Institute si sia adoperato in buona fede per garantire che le informazioni e le istruzioni contenute in questa opera siano accurate, Linux Professional Institute declina ogni responsabilità per errori od omissioni, inclusa, senza limitazione, la responsabilità per danni derivanti dall’uso di questa opera. L’utilizzo di informazioni e istruzioni contenute in questa opera è a proprio rischio. Se qualche esempio di codice o tecnologia che questa opera contiene o descrive è soggetto a licenze open source o è sotto diritti di proprietà intellettuale di terzi, è tua responsabilità assicurarti che se ne faccia uso rispettando tali licenze e / o diritti.

I materiali didattici LPI (Learning Materials) sono un’iniziativa Linux Professional Istitute (https://lpi.org). Materiali didattici e loro traduzioni sono su https://learning.lpi.org.

Per domande e commenti scrivi una mail a: learning@lpi.org.